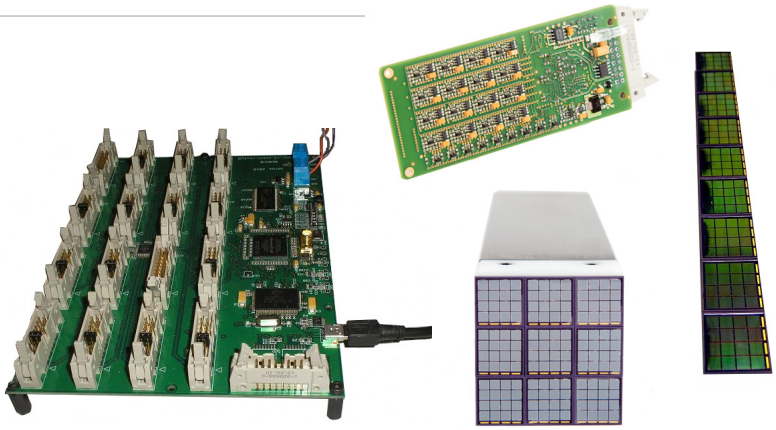
Matrix Readout System for Nuclear Medicine Detection



Document Overview

This document provides the user with a comprehensive description of the hardware and software provided in the SensL Matrix readout system, including system description, detailed design features, software GUI and DLL driver description. The document is split up into the following sections:

OVERVIEW

This section gives a brief description of the Matrix system outlining its main features and applications.

GETTING STARTED

This section provides instructions for unpacking and installing the SensL Matrix hardware and GUI software ready for use with a host computer.

MATRIX GUI SOFTWARE USER GUIDE

Here the controls for all system operations are described, including setup, configuration, data readout, energy spectra plotting and energy resolution calculation.

MATRIX DETAILED SYSTEM DESCRIPTION

This section describes in detail the hardware operation of the system. This includes block diagrams and schematics describing how photon events are detected and the process by which these events are passed through the system and collected by the host PC. This section also includes a detailed description of the configuration process and the various USB registers used for interfacing with the host PC.

MATRIX SOFTWARE DRIVERS

The Matrix system comes with a set of drivers for controlling the hardware within a user application. The Matrix DLL library provides a comprehensive set of functions that allow the user to program the system to carry out various operations with relative ease.

SensL © 2012

Contents

Document Overview 1
Overview 1
Getting Started 1
Matrix GUI Software User Guide 1
Matrix Detailed System Description 1
Matrix Software Drivers 1
System Overview 4
System components 4
Getting Started 6
Help and support 6
Unpacking the System 6
Safety Considerations 7
System Characteristics and Specifications 7
Dimensions 7
Weight 7
Power 7
Temperature 7
System Requirements 7
System Installation Procedures 8
Matrix GUI Software User Guide 9
Array and Pixel Positions 9
General Controls and Status 11
Matrix Channel Configuration 12
EVB Readout Control 15
Energy Plot 17
Timing Plot 17
Plot Select 18
Plot Range 19

Smoothing 19
Energy Resolution Calculation 19
Timing 20
Timing Energy Windows 20
Matrix Detailed System Description 21
Matrix Readout Module 21
Passive Sensor Head 21
Readout Electronics 23
Matrix-EVB Board 29
USB Hardware Registers 30
FIFO Memory 33
Readout FIFO Data Format 33
Matrix Software Drivers 40
DLL Driver Functions 40
Conventions 40
Driver Descriptions 41
Example 52
Appendix 55
Timing Data Processing 55
Sort Coincident Pairs 55
Time-walk Correction 55

System Overview

SensL's Matrix system is the first modular, turnkey readout system specifically designed for nuclear medicine imaging applications. Building on SensL's silicon photomultiplier (SiPM) technology, the MatrixSM-9, MatrixSM-9X1 and MatrixB-9X1 sensor heads provide a fully solid-state, four-side scalable sensor for the readout of LYSO* or BGO* scintillator arrays. The Matrix system integrates the sensor head with all of the electronics required to localize, time stamp and discriminate scintillation events. Digitized event data comprising time, location and energy are sent to the host system via high speed USB interface. By connecting multiple sensor readout modules to a host communications board (the Matrix-EVB), the Matrix system can perform temporal coincidence analysis thereby reducing data rates to the host PC.

The Matrix sensor heads require low power and operating voltage, are compact, highly robust and offer excellent spatial segmentation and uniformity of response. They address the need for high-resolution imagers such as those required for small animal, pre-clinical and mammography PET. The Matrix system's high degree of modularity makes it a compelling and cost effective substitute for the current PMT based systems in full-body PET.

A crosswire readout architecture is used to simplify the readout electronics and to enable 144 sensor channels to be serviced by just 25 readout channels. The design requires the availability of a fast trigger signal, allowing for precise time stamping for coincidence-based systems.

** The MatrixSM heads are designed to work with LYSO while the MatrixB is designed to work with BGO.*

SYSTEM COMPONENTS

The Matrix system architecture is modular and is based on three principal sub-systems:

1. **Sensor Head** - a 12 x 12 pixel (MatrixSM-9) or 4 x 36 pixel (MatrixSM-9X1, or MatrixB-9X1) SiPM array, integrated on a four-side scalable daughter board. The sensor head is fitted with mechanical mounting points and connects via an 80-way flexible printed circuit cable to the front-end electronics board. The two types of sensor head are summarized in Table 1 below.

   The MatrixSM sensor heads are optimized for use with LSO/LYSO scintillators, and the MatrixB for use with BGO scintillator. The MatrixB uses the reverse polarity bias, has a higher amplifier gain and a slower shaping time with respect to the MatrixSM systems.

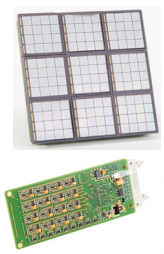
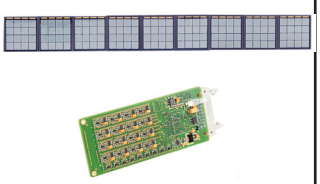
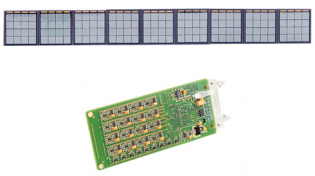
				
Part number	**MatrixSM-9-30035**	**MatrixSM-9-30035-OEM**	**MatrixSM-9X1-30035**	**MatrixB-9X1-30035-SCB**
Format	Square	Square	Linear	Linear
Pixel layout	12 x 12	12 x 12	4 x 36	4 x 36
Enclosure?	Yes	No	No	No

Table 1 Details of the possible readout module options.

2. The **Front-end Electronics** Board serves to bias the 144 SiPMs and amplify their signals. Candidate signals are flagged through a programmable threshold discriminator and are digitized, time stamped, buffered and subsequently read out to the Matrix-EVB Board through a high speed serial interface.

The combination of Sensor Head and the Front-end Electronics Board form the sensor **Readout Module**.

3. The **Matrix-EVB** Communications Board (Matrix-EVB-16) provides all of the necessary control for up to 16 Matrix sensor Readout Modules. It provides a common clock, retrieves and buffers event data and interfaces with the host computer through a high speed USB connection.

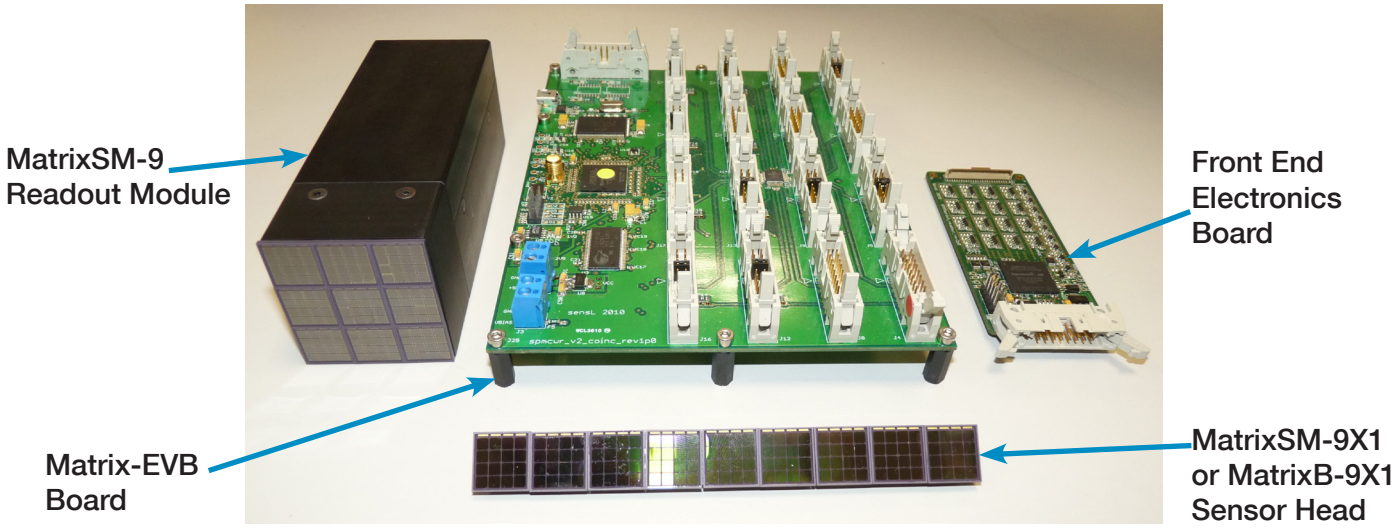


Figure 1 The Matrix System components. A working system will consist of a Matrix-EVB board and up to 16 sensor Readout Modules.

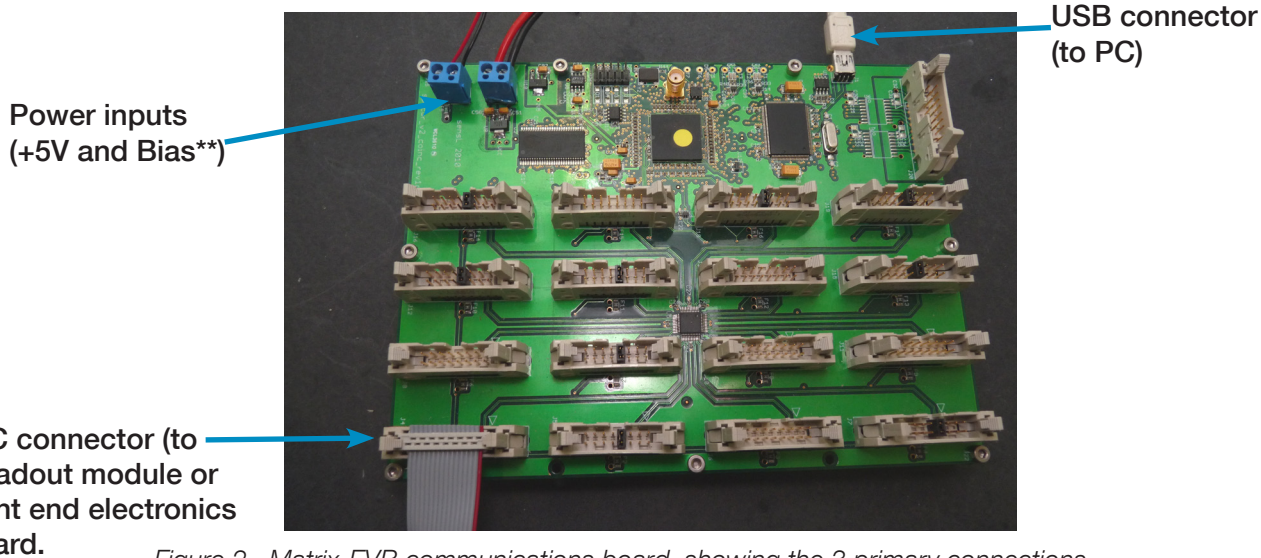


Figure 2 Matrix-EVB communications board, showing the 3 primary connections.

*** Apply **+BIAS** for the MatrixSM sensor heads, and **-BIAS** for the MatrixB sensor heads.*

Getting Started

HELP AND SUPPORT

The most up to date versions of the Matrix system software, firmware and DLLs can all be found on the Matrix support page,

http://sensl.com/matrix9software/

A number of instructional videos related to the use of the Matrix9 can also be found on the website,

http://sensl.com/documentation/videom9/

For all other enquiries, please contact

support@sensl.com

The FPGAs on the EVB board and front-end electronics board can be upgraded without the need to send it back to SensL. This allows the user to benefit from the most recent updates with minimal inconvenience. Please contact SensL for details.

UNPACKING THE SYSTEM

Contents of the package:

- 1 (minimum) Matrix Readout Module (MatrixSM-9-30035)

  or

- 1 (minimum) Matrix Sensor Head (MatrixSM-9X1 or MatrixB-9X1) + Front End Electronics Board
- 1 Matrix-EVB communications board (Matrix-EVB-16)
- EVB board to Readout Module 16-way IDC cable
- USB cable

Note: It is important to install the Matrix software BEFORE connecting the system to the PC for the first time.

SAFETY CONSIDERATIONS

1. The system is not intended for outdoor use
2. Liquids should not be spilled on or into the system.

SYSTEM CHARACTERISTICS AND SPECIFICATIONS

Dimensions

Matrix-EVB Board: 141.0 x 183.0 mm^2
Readout Module: 48.0 x 46.5 x 134.3 mm^3

Weight

Readout Module: 375g

Power

Main Power: +4V - +5.5V : 1A (Matrix-EVB Board) + 0.5A per Readout Module
Bias:

Sensor Head	Bias Range	Current
MatrixSM-9 MatrixSM-9X1	+35V to +40V	5mA per Readout Module
MatrixB-9X1	-35V to -40V	

Table 2 Bias polarities used for different sensor heads

Temperature

Operating: +20°C to +30°C
Storage: -20°C to +70°C

System Requirements

The computer used for the Matrix software requires the following minimum configuration:

- Windows XP, Windows 7 32-bit or Windows 7 64-bit
- 1 GByte of RAM
- At least one spare High Speed USB 2.0 port
- Internet connection (only for downloading the GUI software for first install - see page 8)

SYSTEM INSTALLATION PROCEDURES

Before connecting the system to the PC for the first time, it is very important that the software is installed using the following steps.

1. Download the 'Release Note' PDF and the 'Matrix9 Software' EXE from http://sensl.com/matrix9software/

2. Run the executable **Matrix9_Install_XpY.exe,** (where X.Y is the latest software revision number) by double-clicking on the file. This will start a process that includes the installation of the Matrix9 GUI and drivers as well as the DPlot graphics package and QuickUSB drivers. A number of screens will appear as the installation progresses. Follow the instructions as necessary. There is the option to select file locations and which shortcuts to add. At the end of this process, the software is ready for use.

3. Connect the USB cable supplied between the Matrix-EVB board and the PC*.

4. Connect readout module(s) to the Matrix-EVB using the IDC cable(s).*

5. Power up the Matrix-EVB by applying +5V and the Bias Voltage (see Table 2) to the DC to the screw terminals. There is clear labeling by each terminal to show where each supply should be connected.

6. The PC will recognize that new hardware has been added. Depending on the operating system the drivers may be located automatically, or it may be necessary to select them manually by directing the PC to the directory c:\Program Files\SensL\Matrix9\QuickUSB where the necessary file will be located.

7. Now launch the GUI by running **Matrix9.exe**. Assuming you have installed the Matrix software in the default folder you will find **Matrix9.exe** in c:\Program Files\SensL\Matrix9.

8. The system is now ready for use.

The hardware set-up is demonstrated in a video on SensL's website,

http://sensl.com/documentation/videom9/

Note: Never connect or disconnect readout modules without first switching off the power to the Matrix-EVB board.

Matrix GUI Software User Guide

The Matrix GUI software is designed to allow the user to evaluate the Matrix system. The software provides for:

- Easy setup and configuration of the system
- Streaming of event data, in real time, from FIFO to PC file
- Graphical pixel intensity plotting for imaging
- Easy plotting of energy spectra and calculation of FWHM energy resolution

ARRAY AND PIXEL POSITIONS

Each of the 144 individual SiPM in a given Matrix sensor head can be defined by 2 values, PIXEL (0-15) and ARRAY (0-8).

The value of ARRAY defines the particular ARRAY (0 to 8) in the Matrix sensor. The positioning of the arrays is given for the two types of sensor head (12 x 12 or 4 x 36 pixel layout) in Figures 3 and 4 below.

The value of PIXEL defines the particular SiPM (0 - 15) in the ARRAY. The positioning of the pixels is given for the two types of sensor head (12 x 12 or 4 x 36 pixel layout) in Figures 3 and 4 below.

Figure 3 and 4 below show the view is as if looking at the Matrix sensor head from the front (sensitive) side of the sensor.

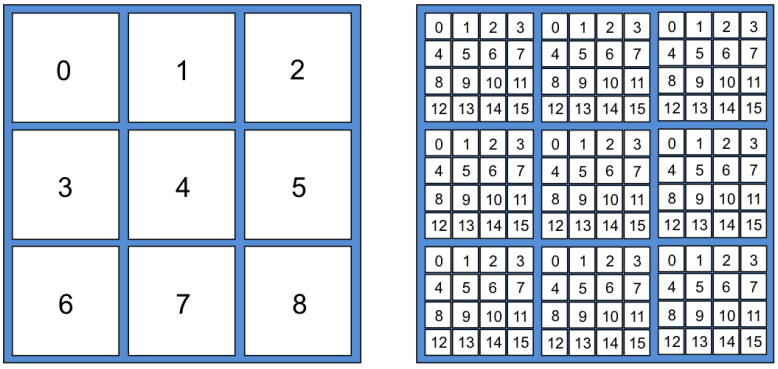


Figure 3 showing the array and pixel locations for the MatrixSM-9

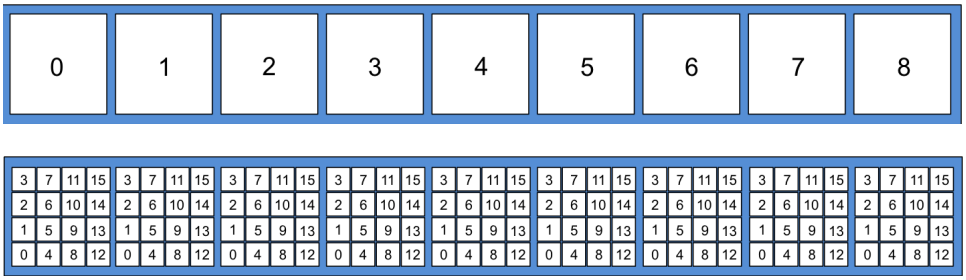


Figure 4 showing the array and pixel locations for the MatrixSM-9X1 and MatrixB-9X1

Main Page

When the Matrix GUI is launched the MAIN Dialog Page is displayed, as shown in Figure 5.

The software allows the use of different sensor heads.

Each of the 16 channels has a 'Linear Array' checkbox that allows the user to inform the readout module electronics if the MatrixSM-9X1 or MatrixB-9X1 type of sensor head (4 x 36 pixel configuration) is used. If the MatrixSM-9 is used, the check box should be left blank. The readout module requires this information to determine the neighboring pixels for region of interest readout. This checkbox also determines how the GUI displays the pixel format.

When a given channel has been set to 'Linear Array' via the checkbox, press 'Save' to save this configuration for that channel. To view the pixel map for his channel, use the 'View' thumbwheel in the bottom-right portion of the MAIN GUI window. It will appear as in Figure 6.

The bias polarity can also be selected, depending on which sensor head is used. See the next page for details.

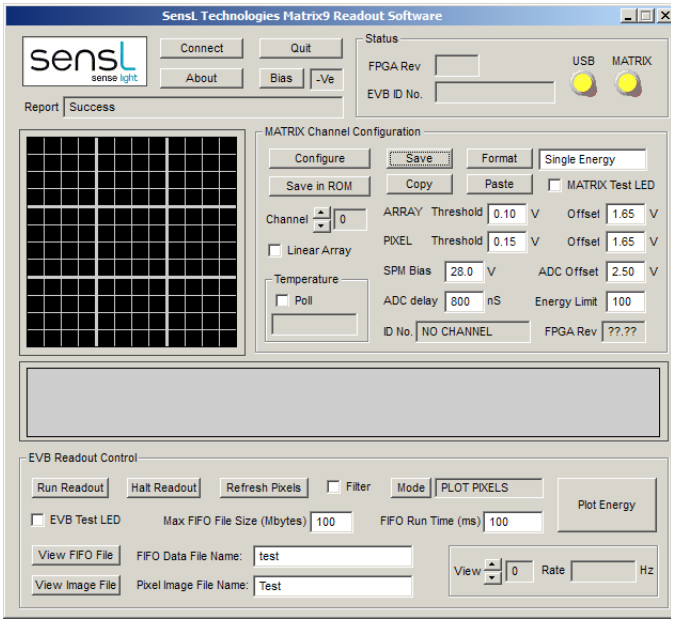


Figure 5 Main GUI page showing the default MatrixSM-9 selection ('Linear Array' unchecked)

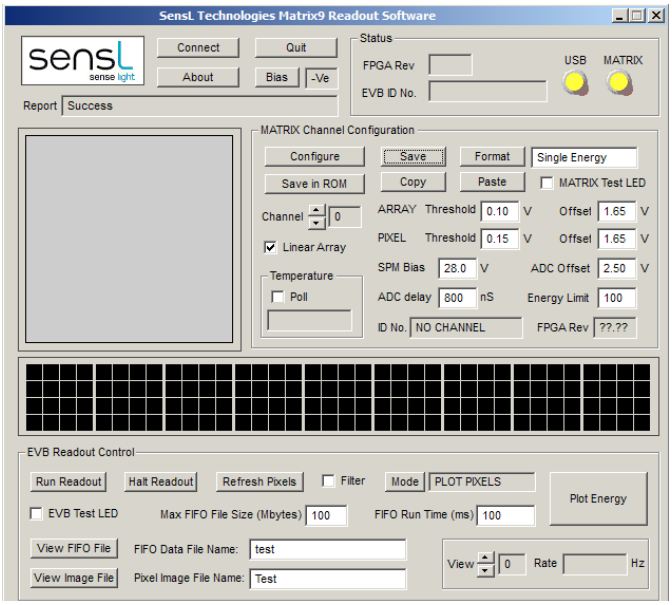


Figure 6 Main GUI page showing the MatrixSM-9X1 & MatrixB-9X1 selection ('Linear Array' checked)

General Controls and Status

Connect

With the USB cable connected and the Matrix system power on, pressing this button will direct the software to search for the Matrix-EVB board and establish communication. Once the communication link is established the USB and MATRIX LEDs should turn green and the FPGA firmware revision number and Matrix-EVB ID number will be displayed.

About

This will display the version of the Matrix GUI software installed.

Bias

Pressing this button toggles between positive (+Ve) and negative (-Ve) bias selections. Positive is used for MatrixSM sensor heads, and negative bias is used for MatrixB sensor heads. The bias polarity should be selected before any of the individual channels are configured.

Quit

Click this button to save current settings and close down the software.

FPGA Rev

When communication between the Matrix-EVB board and the PC is established, this text box will display the FPGA revision number of the Matrix-EVB.

EVB ID No.

When communication between the Matrix-EVB and the PC is established, this text box will display the unique ID number of the Matrix-EVB.

USB and MATRIX LEDS

When communication between the Matrix-EVB and the PC is established, these LEDs will turn GREEN. The MATRIX LED will turn RED if the MATRIX is acquiring data. The USB LED will turn RED if the USB is streaming data from the FIFO to the PC.

Matrix Channel Configuration

Temperature

With the Poll Temperature checkbox checked, this text box will be continually updated with the temperature of the sensor head.

Channel

Each of the 16 Matrix-EVB channels can be selected using this thumbwheel. If the selected channel has a valid Readout Module connected, the FPGA firmware revision number (FPGA REV) and unique ID number of the module (ID No.) will be displayed. A global channel called ALL can be selected to allow the software to simultaneously set and configure all modules.

Configure

This button commands the GUI to configure the selected channel with the saved parameters.

Save in ROM

This button commands the GUI to store the parameters in the module's non-volatile ROM. At power-up, the FPGA in the module will use these ROM values to initialize the module. This allows an optimized set of parameters to be applied to each module in order to avoid having to manually configure at each power-up. Users must keep a note of any values that they save to the ROM in this way, as they are not displayed in the GUI fields at power up.

Table 3 below gives some suggested values for the two sensor head types, for use with LSO/LYSO and BGO respectively.

Parameter	Suggested Values	
	MatrixSM-9/X1	MatrixB-9X1
Array Threshold	0.3V	0.1V
Array Offset	1.65V	1.65V
Pixel Threshold	0.3V	0.15V
Pixel Offset	1.65V	1.65V
SiPM Bias	29.0V	-28V
ADC Offset	1.0V	2.5V
ADC Delay	520ns	800ns

Table 3 Suggested values for the various system parameters to be stored in the ROM

The values given in Table 3 should give reasonable results when the module is used with 511keV gamma rays and LSO/LYSO (MatrixSM-9/X1) or BGO (MatrixB-9X1). However, since all set-ups vary slightly, the user should optimize the values as they apply to their own situation.

Array Threshold and Offset

The discriminator threshold for the 9 ARRAY signal channels can be programmed using a 12-bit DAC. The 12-bit DAC range allows values from 0 to 4095 corresponding to a voltage range of 0 to 3.3V. A second DAC is used to define a DC offset in order to raise the signal above any potential system noise. It is the difference between these 2 DAC values that defines the effective threshold level for the ARRAY channels.

The readout electronics run from a single supply. The offset is usually set to 1.65V (3.3V / 2) to operate the amplifier and comparator input at around the half rail voltage. The Matrix GUI simplifies the setup by allowing the user to define an offset and a threshold relative to this offset in Volts. The software then automatically uses these Voltages to set the ARRAY threshold and offset DAC values. The different sensor heads have different signal polarities as given in Table 4, so the way in which the effective ARRAY threshold level is calculated will differ for each type. For the MatrixSM sensors, see Example 1, and for the MatrixB sensors, see Example 2.

Pixel Threshold and Offset

The discriminator threshold for the 16 PIXEL signal channels can be programmed using a 12-bit DAC. The 12-bit DAC range allows values from 0 to 4095 corresponding to a voltage range of 0V to 3.3V. A second DAC is used to define a DC offset, in order to raise the signal above any potential system noise. It is the difference between these 2 DAC values that defines the effective threshold level for the PIXEL channels.

The Matrix GUI simplifies the setup by allowing the user to define an offset and a threshold relative to this offset in Volts. The software then automatically uses these voltages to set the PIXEL threshold and offset DAC values. The different sensor heads have different signal polarities as given in Table 4, so the way in which the effective PIXEL threshold level is calculated will differ for each type. For the MatrixSM sensors, see Example 2, and for the MatrixB sensors, see Example 1.

Example 1

Offset = 2048 = 1.65V
Threshold = 2480 = 2V

Effective threshold level
= 2V – 1.65V = 0.35V

Example 2

Offset = 2048 = 1.65V
Threshold = 1614 = 1.3V

Effective threshold level
= 1.65V – 1.3V = 0.35V

Sensor Head	Signal Polarity
MatrixSM-9 MatrixSM-9X1	Negative
MatrixB-9X1	Positive

Table 4 Signal polarities

FPGA Rev

The channel FPGA revision textbox will display: **(x) yy.zz**

The **'x'** denotes either a '1' for MatrixSM systems or '2' for MatrixB systems and the '**yy.zz'** is the revision number.

Examples: **(1) 01.05** - MatrixSM at Rev. 01.05
(2) 01.01 - MatrixB at Rev. 01.01

SiPM Bias

The SiPM bias is programmed by a 12-bit DAC. The gain of the BIAS interface is dependent on the Matrix sensor head used:

Sensor Head	DAC Gain
MatrixSM-9 MatrixSM-9X1	10.478mV per bit
MatrixB-9X1	8.801mV per bit

Table 5 DAC gain for bias setting

Example: DAC value = 1718

BIAS Voltage = 1718 x 10.478mV = 18V

The Matrix GUI simplifies this setup by allowing the user to define the BIAS in volts. The GUI translates this voltage into the corresponding DAC value.

ADC Offset

This 12-bit DAC sets the DC offset for the differential input signal for the ADCs. Its full range is 0V – 3.3V.

ADC Delay

This value selects the time duration for extracting the energy value from the output of the shaper circuit (shaping time of 0.5µs). An 8-bit DAC allows values from 0µs – 2.55µs, with each bit being equal to 10ns.

Energy Limit

This value defines the minimum energy allowable for an event to be regarded as true. This value is input as an ADC bin number. When an event is detected, the ADC value for the pixel is compared with this value. If it is less than the Energy Limit value the event will be discarded. This is used to filter out undesirable noise related events and hence improve the overall event bandwidth.

Matrix Test LED

This checkbox is used to define if the sensor readout module LED (see Figure 9, back view of the Readout Module) is on or off. This is particularly useful for verifying communication or identifying a particular module.

Linear Array

Check this box when one of the 'Linear Arrays' are used, either the MatrixB-9X1 or the MatrixSM-9X1.

Format

This button is used to select one of three modes of operation:

- SINGLE ENERGY
- REGION OF INTEREST
- ALL ENERGIES

SINGLE ENERGY

In this mode the module detects an event and then saves the TDC value, pixel location and energy value.

REGION OF INTEREST

In this mode the module detects an event and then saves the TDC value, pixel location and energy of the pixel followed by the energy of all 8 surrounding pixels.

ALL ENERGIES

In this mode the module detects an event and then saves the TDC value, pixel location and all 16 energies from the array.

Copy and Paste

These 2 buttons allow the user to copy the parameters of a selected channel and then paste these values into another channel.

Save

This button is used to save all the parameters of the selected channel ready for storing to the module for later use via the Configure button. Each channel can have its own stored values.

Note: When optimized values have been determined for a given module, the Save to ROM button (see above) can be used instead, which stores the parameters in the modules own EEPROM and automatically applies these saved values upon power-up, avoiding the necessity to manually configure.

EVB Readout Control

Mode Button

The mode button is used to select the mode of operation for the main page. This can be set to:

STREAM TO FILE

This mode is used to stream FIFO data continuously to a file. When the Run Readout button is pressed, the system will stream all event data from the FIFO to a file named in the **FIFO Data File Name** text box with the suffix **RAW**. The data will be acquired for a time that is defined in the FIFO Run Time (ms) text box. The user can also define the maximum allowable size for this file using the Max FIFO File Size (Mbytes) text box. Therefore acquisition will stop after the programmed runtime or the maximum file size is reached.

The format of the resulting binary **RAW** file is the same as described on pages 34 - 35. For example, if the streaming was carried out in Single Mode (Mode 0) then the events will comprise of 4 x 16 bit words as described on page 34 ‘Mode 0’.

A text form of the **RAW** file can be generated by pressing the View FIFO File button. This converts the **RAW** file into a text format with the suffix **FFO** and then launches a text editor that displays the values of the

file. Alternatively the user can locate the file using the PC browser and open the file using a spreadsheet application such as EXCEL. These files will be saved in the folder ***ROOT\data*** where ROOT is the folder the Matrix GUI resides in.

Note: The **RAW** file is in binary format and is therefore relatively small when compared with the **FFO** file. The use of **FFO** files is mainly intended for viewing short runtimes. If post processing is to be carried out on long runtimes then it is recommended that the user works with the **RAW** file.

PLOT PIXELS

This mode will continually read events and keep a running count of the number of valid events for each pixel. The image will be continually updated with a grey scale based on the event count of each pixel. The pixel with the most events will be white. All other pixels will be a grey scale based on the percentage of counts compared to the pixel with the highest count. The image can be refreshed at any time by pressing the Refresh Pixels button. A refresh will clear all the counts putting all pixels back to black. When the Plot Pixels process is halted the counts for each pixel are saved in an image file defined by the Pixel Image File Name text box. This file can be viewed through the GUI using the View Image File button, which launches a text editor that displays the values of the file. Alternatively the user can locate the file using the PC browser and open the file using a spreadsheet application such as EXCEL. These files will be saved in the folder ***ROOT\data*** where ROOT is the folder the Matrix GUI resides in.

Filter Checkbox

The filter must be used for detecting coincidence pairs and making timing measurements. If this checkbox is not checked then the Matrix-EVB will pass all events to the FIFO for streaming to the PC. If this is checked then the Matrix-EVB will carry out basic filtering of the data to remove obvious single events. Obvious singles are events that occur with dead times both before and after that are greater than 40ns. The result is a reduction in event traffic with all data events being potential members of a coincidence pair. The user can use the TDC values to do further filtering to determine true coincidence pairs.

Run Readout

This button starts the Matrix system operating in the defined mode.

Halt Readout

This button halts the Matrix system acquisition in either mode.

EVB LED

This checkbox is used to define if the Matrix-EVB board test LED is on or off. This is particularly useful for verifying communication between the PC and the system.

Plot Energy Button

This button is used to launch the Energy Plot interface (See next section) for:

- Viewing energy spectra of selected pixels
- Performing peak fitting to calculate energy resolution
- Plotting coincidence timing histograms

View and Rate

When the system is running in PLOT PIXEL mode the View thumbwheel allows the user to view the pixel image of a particular channel. During acquisition the Rate text box will continually display the event rate of the module in Hz.

ENERGY PLOT

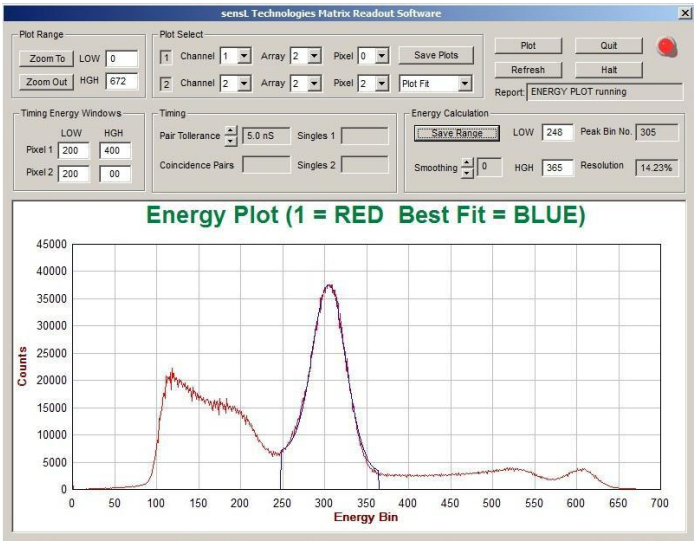


Figure 7 Screenshot of the Energy Plot Page showing the LYSO energy spectra of a selected pixel

The Energy Plot Page, as shown in Figure 7, allows the user to acquire and plot energy spectra. When an event is read from the FIFO, the memory location corresponding to the energy value for a given pixel in a given module is incremented. The energy value is in the range of 0 to 4095 (12-bit ADC). The Matrix GUI allows up to 2 plots to be viewed simultaneously. The channel, array and pixel buttons are used to select the particular pixel.

TIMING PLOT

The Energy Plot Page also allows the user to plot the timing between coincidence pairs detected on two selected pixels of two independent channels. The channel, array and pixel buttons are used to select the particular pixel.

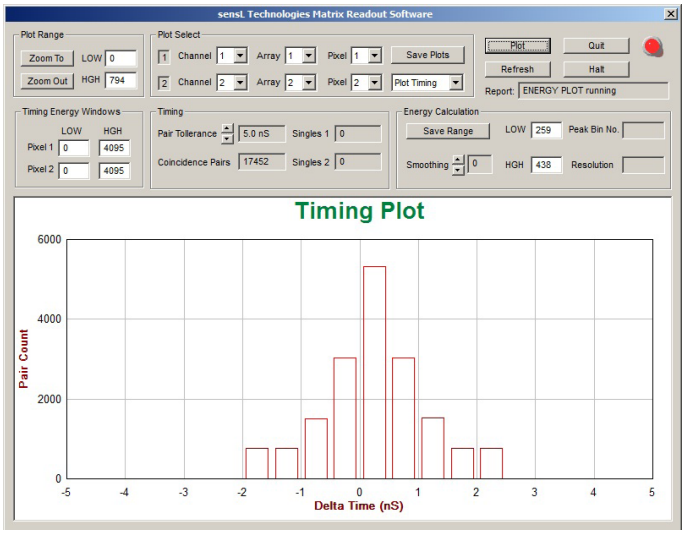


Figure 8 Screenshot of the Energy Plot Page showing the LYSO coincidence timing plot of a selected pixel

TIMING OUTPUT FILE

As the '**Timing Plot**' is a real-time plot, the analysis to determine the true coincidence pairs is somewhat limited. Therefore, the plot should only be regarded as an estimated result. At the same time as the real time plot is progressing, all events are stored to a binary file in the ***ROOT\data*** folder that can be used to post process the data to make a better measurement of the CRT. This binary file will have the same name as appears in the '**FIFO Data File Name**' text box on the main GUI page with the suffix **TIM**.

The format of the resulting binary **TIM** file is the same as described on page 34, 'Mode 0'. Each event is stored as 4 x 16 bit words.

A technique for post processing this data to get the best timing results is described in the Appendix.

General Controls

Plot
Pressing this button will start the software acquiring data and saving the binned energy values to the PC.

Refresh
Pressing this button will, at any time, clear all the bins.

Halt
Pressing this button will halt the acquisition.

Quit
Click this button to close down the Energy Plot window.

Plot Select

Pixel 1, 2
The GUI allows the energy spectra of up to 2 pixels to be plotted simultaneously. The Channel, Array and Pixel dropdown boxes allow the user to specify 2 particular pixels in the system.

Plot Type
The Plot type can be selected as:

Pixel 1:	Plot of pixel energy spectrum defined by Pixel 1 buttons.
Pixel 2:	Plot of pixel energy spectrum defined by Pixel 2 buttons.
Both:	Plot of both pixel energy spectra simultaneously.
Fit:	Plot best fit for Pixel 1. See Energy Resolution Calculation, page 19.
Timing:	Plot the coincidence timing data for Pixel 1 and 2 as a histogram.

Save Plots

This button will save the plot data (of any of the Plot Types listed above) for the 2 pixels in files of name:

Cxx_Ayy_Pzz.plt

Where xx = Channel number 0-15

yy = Array number 0-8

zz = Pixel number 0-15

These files will be saved in the folder ROOT\data where ROOT is the folder the Matrix GUI resides in.

Plot Range

The Plot Range section is used to zoom in and out of the energy spectrum. The X-axis of the graph is by default set to the range of 0-1000 but can be set up to a maximum of 0-4095 corresponding to the full range of the 12-bit ADC. To zoom in to a particular region of the graph, set the LOW value to the start of the region of interest and the HIGH value to the end of the region of interest. Clicking the Zoom In button will now make the software plot this specific region. Clicking the Zoom Out button will make the plot cover the entire 4096 bins.

Alternatively the plot range can be defined using the mouse and the shortcuts given in Table 6.

Special Key(s)	Mouse Button	Action	Result
Ctrl	Left Button	Double Click	Zoom extents 0-1000 bins
Shift	Left Button	Double Click	Zoom extents 0-4095 bins
Ctrl	Left Button	Single Click	Define lower range
Ctrl	Right Button	Single Click	Define upper range
Ctrl + Shift	Left Button	Sweep left to right	Define range

Table 6 Shortcuts for defining Plot Range

The zoom features can be operated while the plot process is running.

Smoothing

The Smoothing Factor thumbwheel allows the user to select a smoothing factor for the plot, to allow a modification of the data to diminish outlying data points that may be present in low-count data sets. A given data value x will become the running average of 'N' consecutive, neighboring data points where 'N' is the smoothing factor. A smoothing value of 0 will disable this feature such that no smoothing takes place.

Energy Resolution Calculation

This section is used to dynamically plot a best fit and calculate the energy resolution of a specific pixel. The Plot Type must be set to FIT mode. When running, the energy plot will be carried out for the pixel defined by Pixel 1. At the same time the software will carry out a best fit plot of a Gaussian curve between the range of bins defined by LOW and HIGH. At any time during the plot, the values of LOW and HIGH can be changed by entering new values into the relevant boxes. Alternatively, the shortcuts given in Table 7 can be used to specify LOW and HIGH. By pressing the Save Range button the software will use the new values to re-calculate and plot the new best fit.

As the plot is running, the software will report the peak bin and calculated energy resolution based on the best fit curve.

Special Key(s)	Mouse Button	Action	Result
Shift	Left Button	Single Click	Define lower FIT (LOW)
Shift	Right Button	Single Click	Define upper FIT (HIGH)
Ctrl + Shift	Right button	Sweep left to right	Define FIT Range

Table 7 Shortcuts for defining FIT range

Timing

Pair Tolerance

This defines the maximum time difference in 0.5ns steps that is allowed for an event between Pixel 1 and Pixel 2 to be considered as a coincidence pair.

Pixel 1, 2 Singles

When running in timing mode, these text boxes will be continually updated with the total number of Pixel 1 and Pixel 2 single events. A single event is an event on a pixel that did not have an associated coincidence event on another pixel.

Coincidence Pairs

When running in timing mode, this text box will be continually updated with the total number of coincidence pairs detected. A coincidence pair is when an event on Pixel 1 and Pixel 2 occurs within the 'Pair Tolerance' time.

When the timing mode is running, a histogram will be plotted of the pairs detected. The X-axis will be a set of bins separated by 0.5ns. The Y-axis will be the number of pairs detected that occurred in that particular bin.

Timing Energy Windows

The timing energy windows are used to carry out further filtering of the event data. To determine true timing it is required to only use coincidence pairs where the energy of each event occurred within the photo-peak window. To determine the settings of these values it is first necessary to carry out an energy spectra plot. From this plot the photo-peak window can be determined. For example, in the previous Energy Plot shown in Figure 7, it can be seen that the peak is in the range ~250 to ~370. Therefore, for this pixel the Timing Energy Window should be set to 250 LOW and 370 HGH.

Matrix Detailed System Description

MATRIX READOUT MODULE

The Matrix Readout Module comprises a passive sensor head connected to readout electronics. In the case of the MatrixSM-9, the sensor head and the front end electronics board are housed inside an enclosure, as shown in the Figure 9 below. In the case of the MatrixSM-9X1 and the MatrixB-9X1, the sensor head and front end electronics board are not in an enclosure.

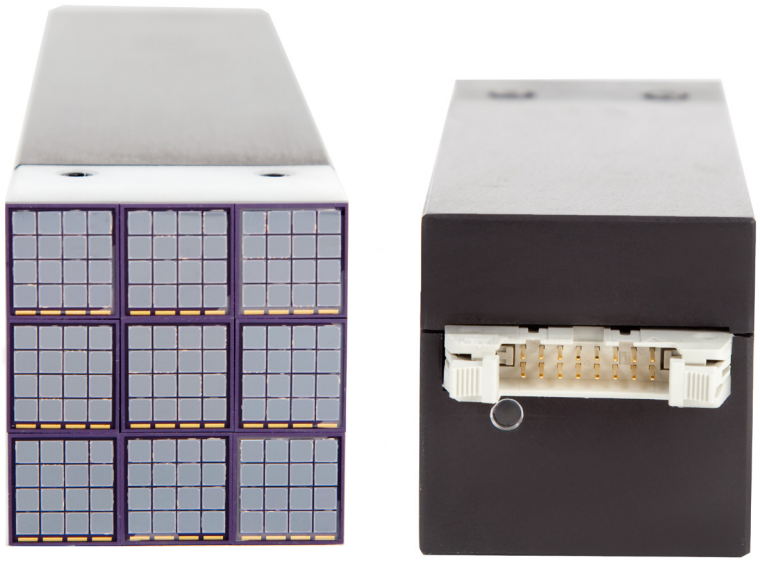


Figure 9 Front and back views of the Readout Module MatrixSM-9

Passive Sensor Head

The sensor heads are made up of nine 16-pixel SiPM array modules forming a 144 pixel array. These arrays are connected together using a "**Scrambled Crosswire Readout**" technique. This technique is used to greatly reduce the number of readout channels whilst still allowing the readout of individual local events for **"Region of Interest"** analysis. To explain the principle, see Figure 10.

In the **MatrixSM** sensor heads, all 16 cathodes of each 16-pixel array are connected together, giving a total of 9 ARRAY event lines from the whole sensor head. The corresponding anode from each array is connected together to provide a total of 16 PIXEL event lines from the whole sensor head.

In the **MatrixB** sensor heads, the set-up is reversed, with all 16 anodes of each 16-pixel array connected together, giving a total of 9 ARRAY event lines. The corresponding cathode from each array is connected to provide a total of 16 PIXEL event lines from the whole sensor head.

When a particular SiPM fires, a pulse will occur on its corresponding ARRAY and PIXEL event lines. These pulses are used by the readout electronics to determine which pixel fired and the amplitude of the PIXEL signal is used to determine the energy. In this way, 144 sensors are read out by just 25 electronic channels.

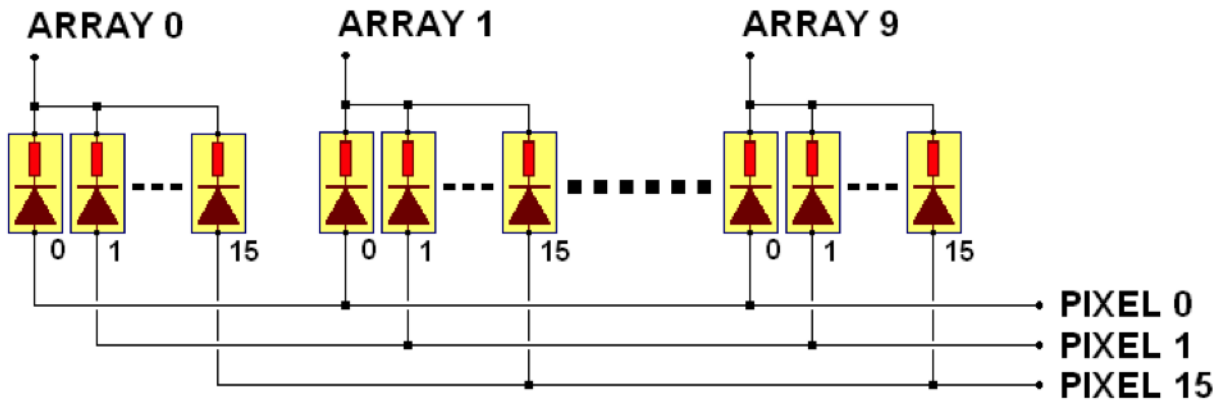


Figure 10 Scrambled Crosswire Readout, shown here for the MatrixSM-9/X1.

Figure 11 shows the positions of the 144 pixels in both types of sensor head. As can be seen, each pixel 0 to 15 is spatially separated by at least 3 pixels. This means that if an event occurs in a particular pixel it can be assumed that the neighboring pixel energies can also be determined. For example, if the pixel event is ARRAY 0, PIXEL 6 then the energy values of the surrounding pixels will be those read from PIXEL 1, 2, 3, 5, 7, 9, 10 and 11. This is based on the reasonable assumption that only 1 event occurs at any time. Therefore when the ARRAY 0, PIXEL 6 event occurs there will be no energy contribution to these lines from other pixels in the other 8 arrays.

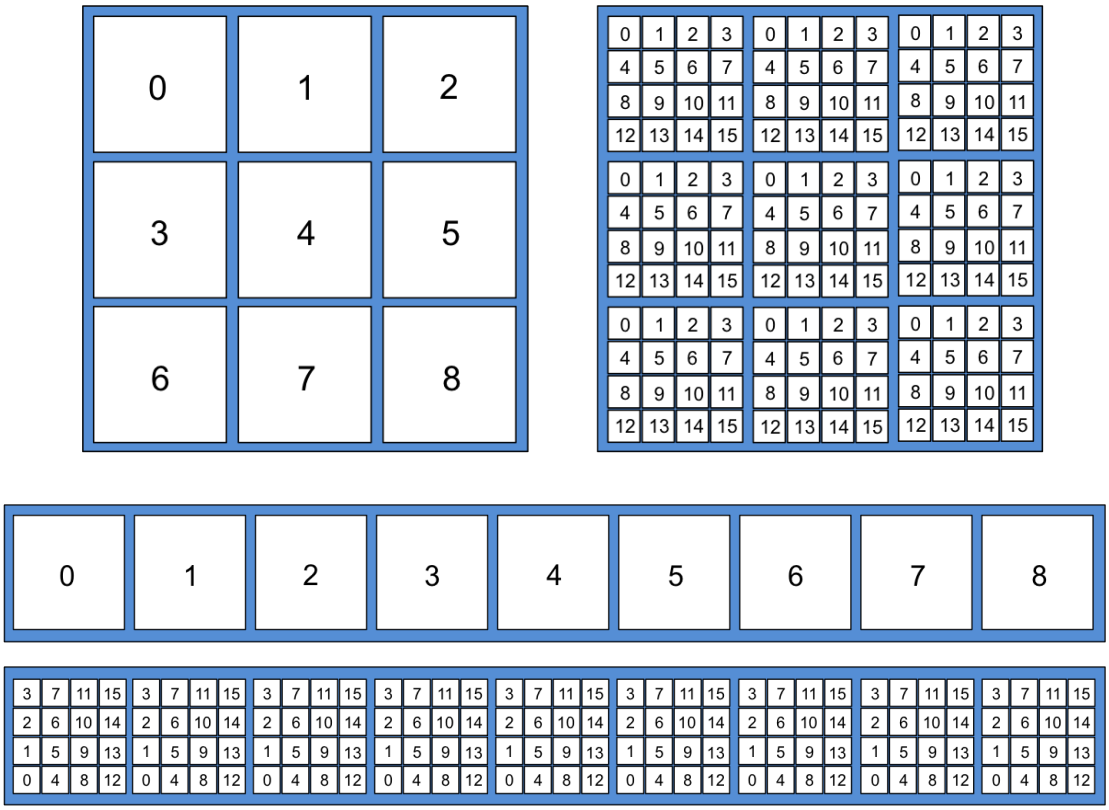


Figure 11 Array and Pixel location map for the MatrixSM-9 (above) and the MatrixSM-9X1 or MatrixB-9X1 (below)

Readout Electronics

Figure 12 shows a block diagram of the readout electronics.

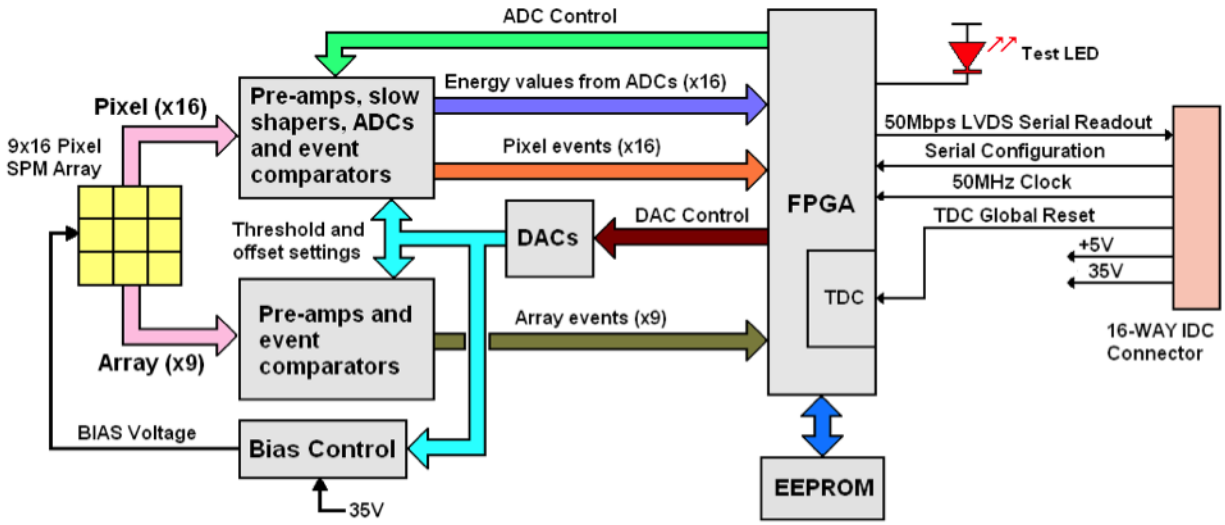


Figure 12 Readout Electronics

The 9 ARRAY event lines are fed from the sensor head to pre-amplifiers before being passed to the input of a fast comparator. The outputs of these 9 comparators are then passed to the FPGA for processing to be used for the event location data.

The 16 PIXEL event lines are fed from the sensor head to pre-amplifiers before being passed to the inputs of a fast comparator and a slow energy shaper. Figure 13 shows a block diagram of the analog section of the readout electronics. Data from the PIXEL event lines are ultimately used to determine event location, energy and timing information.

Each ARRAY and PIXEL event line from the sensor head is passed through a preamplifier. Programmable 12-bit DACS provide a DC offset that lifts the event signal above the ground level. The following comparator has a programmable 12-bit DAC to define the threshold of the trigger. The effective threshold voltage is the difference between the DC offset and the threshold setting. Both off these DACs cover a voltage range of 0 - 3.3V.

The output of the PIXEL preamplifier is also fed to a slow shaper to optimize the signal for energy readout. The output of the slow shaper is connected to a 12-bit, serial ADC. A further programmable DAC voltage is provided to lift the baseline of the ADC to a desired position within the ADC count range.

A further programmable 12-bit DAC is provided to control the BIAS voltage presented to the 144 PIXEL SiPM Array. Each bit corresponds to 10.478mV (MatrixSM) or 8.801mV (MatrixB) of bias up to a maximum of (-)35V (polarity dependent on sensor head type, Table 2).

All of the DAC values are programmed via commands to the FPGA. These DAC values can be saved in EEPROM for automatic configuration at power-up.

The module also has an LED that can be commanded to be ON or OFF. This is particularly useful for module identification/detection when working with a number of modules.

The 16-way connector provides power to the module, configuration and control via a serial data link and a high speed LVDS data bus for streaming readout data to the Matrix-EVB board. The 16-way connector also provides the system clock and global reset signal for synchronizing the TDC.

Note: For details of the serial configuration link see Matrix-EVB board description.

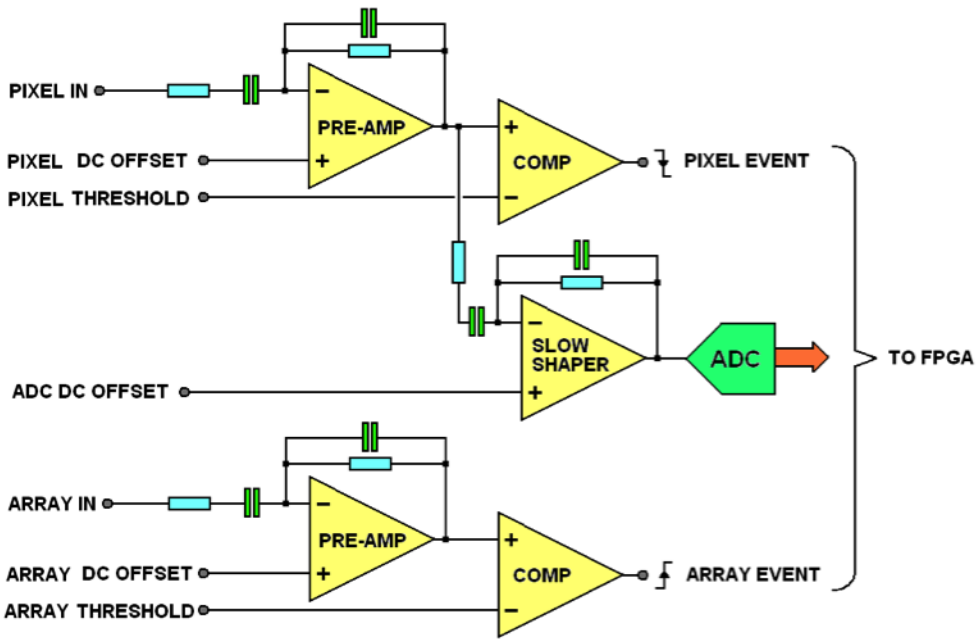


Figure 13 Analog electronics of a Pixel and Array event line

FPGA Readout Procedure

Each event input (9 x ARRAY, 16 x PIXEL) has a latch to hold the occurrence of an event. Due to the inverting of the analog electronics, for PIXELS the event is a HI-LO transition and for ARRAYS the event is a LO-HI transition. The FPGA carries out the readout process as follows:

- When the FPGA receives a command to start the readout process it first clears all event latches and then polls for an event on one of the PIXEL lines.

- The HI-LO transition of the PIXEL event automatically latches the value of a TDC implemented in the FPGA. This TDC is a 16-bit counter with an LSB of 500ps.

- The PIXEL event starts a programmable timer with a resolution of 10ns per bit. When this timer reaches its programmed duration the output signal of all the slow shapers are sampled on the ADCs to provide the 16 PIXEL energy values.

- When all energy values have been read the FPGA interrogates the ARRAY event latches to determine in which array the event occurred. If there is no event on the ARRAY side then the FPGA sets an error bit that is transmitted with the readout data.

- If the energy of the EVENT PIXEL is less than the programmable minimum (Energy Limit) then the event will be ignored.

- Finally, the FPGA clears all the latches for the next event and transmits the event data to the Matrix-EVB board via the 50Mbps LVDS bus. The content of this data will vary depending on the mode of operation. As a minimum the event data will contain the ID (ARRAY, PIXEL), the TDC value and the energy for the pixel that caused the event.

Readout Data Stream Format

The format of the 50Mbps serial data stream is shown in Figure 14.

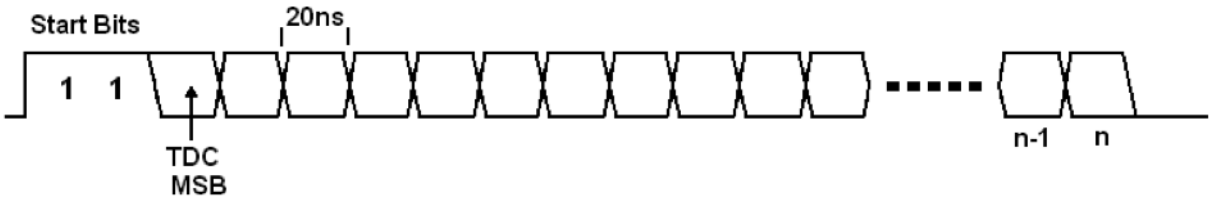


Figure 14 Readout Data Stream

The idle state of the LVDS data line is logic 0. The start of a new data stream is defined by the data line going HI for 2 bit durations (40ns). The following bits define the contents of the data. The first 16 bits of the data is always the TDC value for the event. This is followed by a 16 bit event ID word that defines which pixel fired. The remaining bits describe the energy information. The energy information and hence packet length will vary depending on the mode of operation as described below. The various operation modes are referenced as follows (more detail on the modes can be found on page 15):

Mode 0 = Single Energy

Mode 1 = Region of Interest

Mode 2 = All Energies

Mode 8 = status reporting

MODE 0

After the 2 start bits, the event is represented by 3 x 16-bit data words (MSB transmitted first).

WNo.	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
W1	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
W2	X	1	0	0	M	M	M	M	A	A	A	A	P	P	P	P
W3	0	0	0	0	E	E	E	E	E	E	E	E	E	E	E	E

T: 16-bit TDC value with 500ps LSB

X: This bit is set if the data is invalid. If this bit is '1' the data must be ignored.

M: Operating Mode

A: Array number (0-8)

P: Pixel number (0-15)

E: 12-bit energy value of pixel

MODE 1

After the 2 start bits, the event is represented by 11 x 16-bit data words (MSB transmitted first).

WNo.	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
W1	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
W2	X	1	0	0	M	M	M	M	A	A	A	A	P	P	P	P
W3	H	H	H	H	E0	E0	E0	E0	E0	E0	E0	E0	E0	E0	E0	E0
W4	H	H	H	H	E1	E1	E1	E1	E1	E1	E1	E1	E1	E1	E1	E1
|	|	|	|	|	|	|	|	|	|	|	|	|	|	|	|	|
W11	H	H	H	H	E8	E8	E8	E8	E8	E8	E8	E8	E8	E8	E8	E8

T: 16-bit TDC value with 500ps LSB

X: This bit is set if the data is invalid. If this bit is '1' the data must be ignored.

M: Operating Mode

A: Array number (0-8)

P: Pixel number (0-15)

E0: 12-bit energy value of EVENT Pixel

E1-E8: 12-bit energies from surrounding pixels

H: If the energy isn't an array due to boundary pixel, these bits will be set HI and the energy will always be 0. If the energy is real then these bits will always be 0.

The relationship between energy number and surrounding pixels is as follows:

E1	E2	E3
E8	E0	E4
E7	E6	E5

MODE 2

After the 2 start bits, the event is represented by 18 x 16-bit data words (MSB transmitted first).

WNo.	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
W1	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
W2	X	1	0	0	M	M	M	M	A	A	A	A	P	P	P	P
W3	0	0	0	0	E0	E0	E0	E0	E0	E0	E0	E0	E0	E0	E0	E0
W4	0	0	0	0	E1	E1	E1	E1	E1	E1	E1	E1	E1	E1	E1	E1
|	|	|	|	|	|	|	|	|	|	|	|	|	|	|	|	|
W18	0	0	0	0	E15	E15	E15	E15	E15	E15	E15	E15	E15	E15	E15	E15

T: 16-bit TDC value with 500ps LSB

X: This bit is set if the data is invalid. If this bit is '1' the data must be ignored.

M: Operating Mode

A: Array number (0-8)

P: Pixel number (0-15)

E0-E15: 12-bit energies of all ADCs. The energy values E0 – E15 correspond to the pixel position numbering as given in Figures 3 and 4 on pages 9 and 10.

MODE 8

This MODE is reserved for reading back the READOUT board FPGA REV, ID and TEMPERATURE. When in this mode the READOUT board will send 1 message only. The format of this message is as follows:

Word 0: 4 digit BCD value of the FPGA code revision

Word 1-4: 64-bit board ID

Word 5: Temperature of sensor head

Word 6-15: Null (For future development)

Note: Word 5 is a signed number (16 bits), where 1 = 1/16°C. The table below shows some examples of the value to temperature conversion.

TEMPERATURE	DIGITAL OUTPUT (Binary)	DIGITAL OUTPUT (Hex)
+125°C	0000 0111 1101 0000	07D0h
+85°C*	0000 0101 0101 0000	0550h
+25.0625°C	0000 0001 1001 0001	0191h
+10.125°C	0000 0000 1010 0010	00A2h
+0.5°C	0000 0000 0000 1000	0008h
0°C	0000 0000 0000 0000	0000h
-0.5°C	1111 1111 1111 1000	FFF8h
-10.125°C	1111 1111 0101 1110	FF5Eh
-25.0625°C	1111 1110 0110 1111	FE6Fh
-55°C	1111 1100 1001 0000	FC90h

*The power on reset value of the temperature register is +85°C

For full details please refer to the DS1822 data sheet.

MATRIX-EVB BOARD

The EVB Board is responsible for the configuration of the Readout Modules and the collecting of event data for passing to the host computer. Figure 15 shows a picture of the board identifying the various connectors.

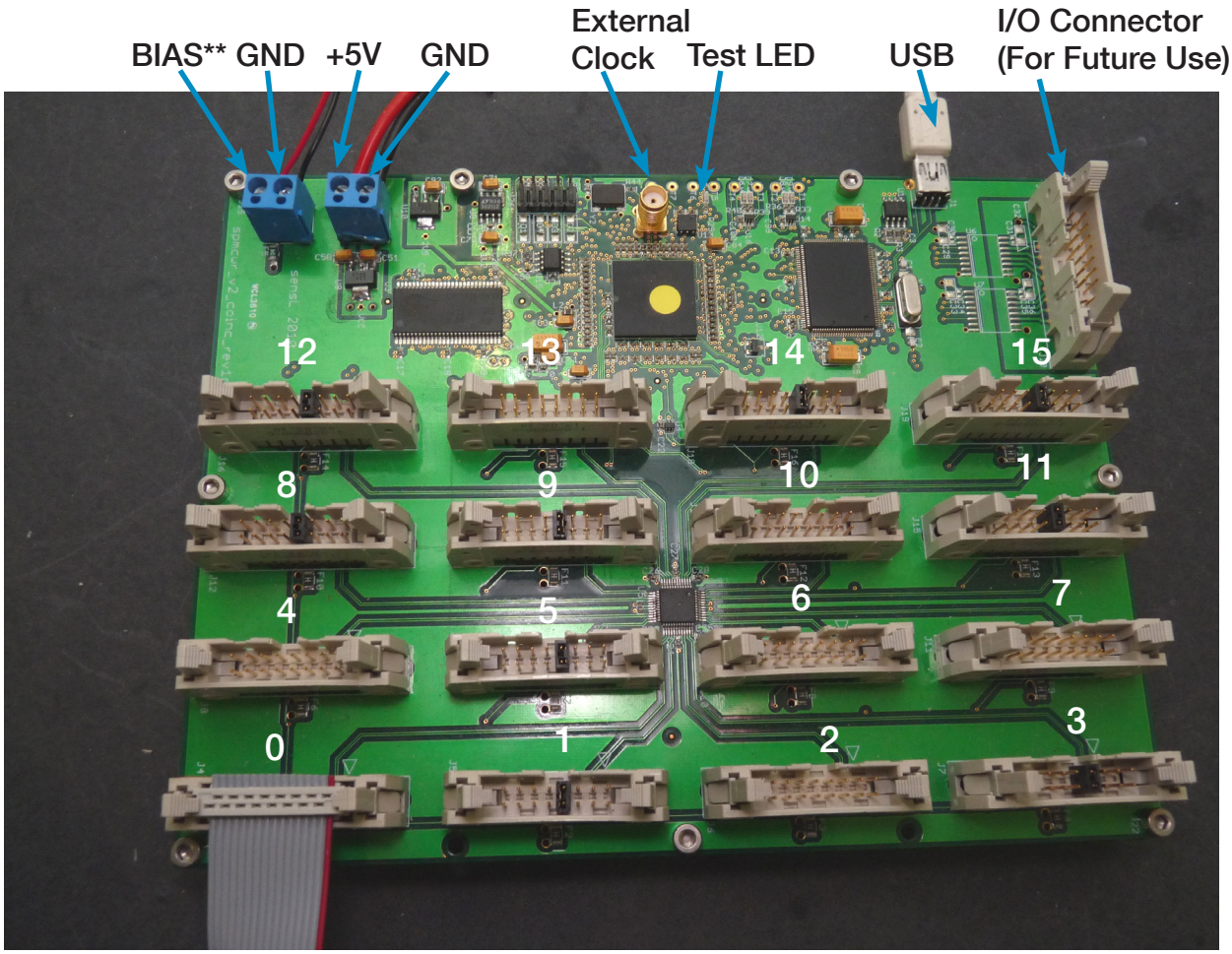


Figure 15 Matrix-EVB Board

The board interfaces with up to 16 Readout modules via 16-pin IDC connectors. The locations of the channels (0-15) are shown in Figure 15. Power (Bias and +5V DC) is supplied to the board via screw terminals. A general purpose I/O connector and External Clock SMA are also provided. See Table 2 for details on the Bias to use.

The power for the Readout Modules is supplied via the 16-pin IDC connectors. The board also provides a common 50MHz clock and TDC Reset and to all channels. Figure 16 shows a block diagram of the EVB Board.

Note: The General Purpose I/O and External Clock SMA connectors are for future use. Do not use these connectors without first consulting SensL.

*** Apply **+BIAS** for the MatrixSM sensor heads, and **-BIAS** for the MatrixB sensor heads.*

The FPGA on the EVB board can be upgraded without the need to send it back to SensL. This allows the user to benefit from the most recent updates with minimal inconvenience. Please contact SensL for details.

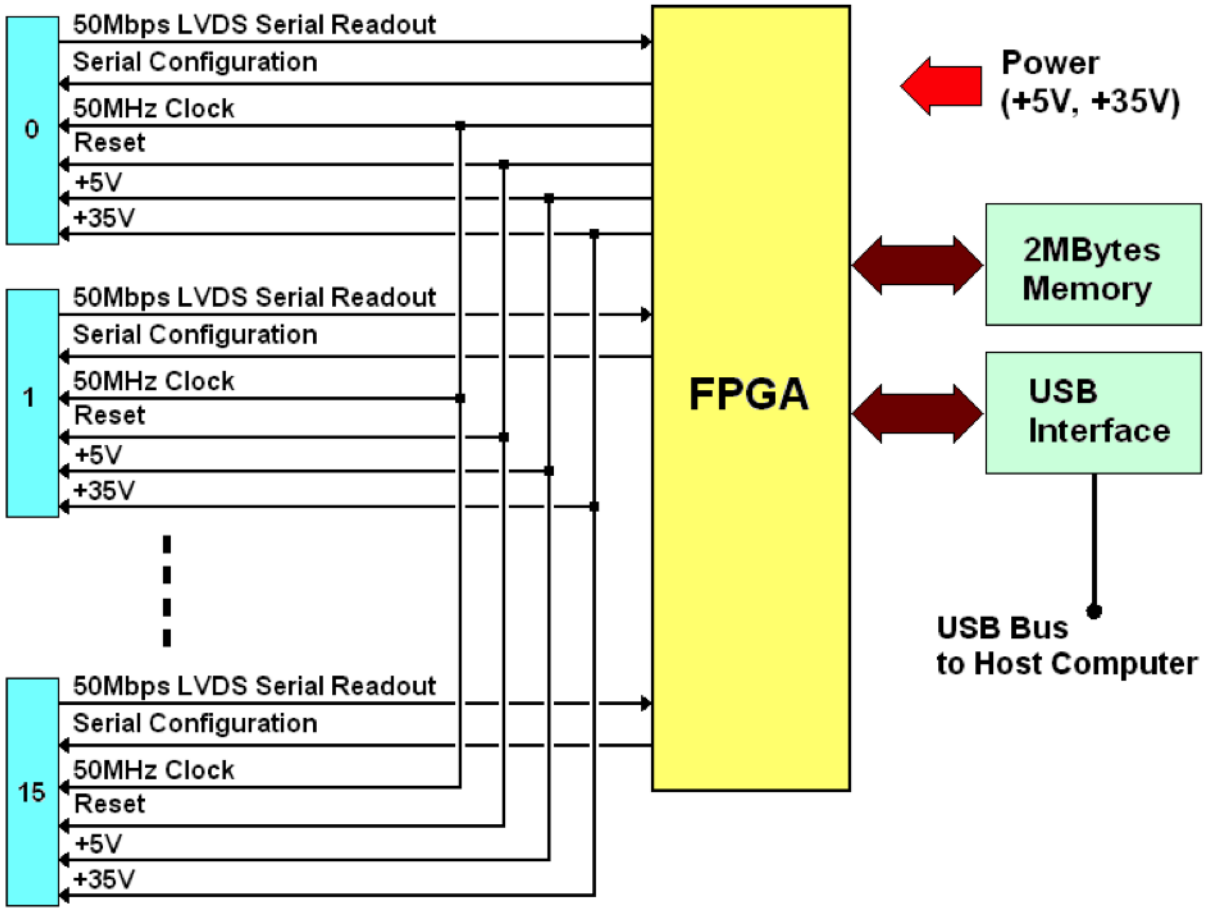


Figure 16 Matrix-EVB Board Block Diagram

USB Hardware Registers

Configuration and readout of the system is carried out via a set of registers residing on the FPGA board. These registers are accessed directly via the USB interface and are used to setup the high speed data streaming interface for real time readout of the data and configuration of the READOUT modules.

SET_UCTRL

Type = Write

GPIDADR = 0

D15-D12	D11-D8	D7-D5	D4	D3	D2	D1	D0
0000	MMMM	000	TR	LD	PR	RR	UR

M: Four bit number defining mode of operation

TR: TDC global reset to READOUT modules (LO = RESET, HI = ACTIVE)

LD: Control of test LED

PR: When set to '1' the memory counter will be reset and ready for a new USB data transfer

RR: Readout state machine reset (active HI)

UR: USB transfer state machine reset (active HI)

SET_RADDC

Type = Write

GPIDADR = 1

D15-D0
Don't care

A write to this register clears the readout memory address counter ready for start of FIFO reading.

SET_SERPT

Type = Write

GPIDADR = 2

D15-D13	D12-D8	D7-D0
000	CCCCC	AAAAAAAA

This register sends ASCII characters to the READOUT modules for configuration.

C: READOUT Module channel number (0-15). If 16 or greater then message will be broadcast to all channels simultaneously.

A: Configuration ASCII character to send to module

SET_UCNTH

Type = Write

GPIDADR = 3

D15-D4	D3-D0
000000000000	UCNT19-16

The least significant 4 bits defines D19 to D16 of the USB memory word count. The USB word count defines how many words are to be transferred from the memory to the USB bus.

SET_UCNTL

Type = Write

GPIDADR = 4

D15-D0
UCNT15-0

This register defines D15 to D0 of the USB memory word count. The USB word count defines how many words are to be transferred from the memory to the USB bus. This register is a 16-word PAGE register. The LSB represents 16 words.

READ_USTAT

Type = Read

GPIDADR = 0

D15-D2	D1	D0
00000000000000	UBSY	MERR

MERR: When set to '1' this bit indicates that the USB read process has not kept up with the READOUT Processor. This bit is cleared by asserting the **PR** bit in the **SET_UCTRL** register.

UBSY: When set to '1' this bit indicates that the USB data transfer processor is busy.

READ_REVR

Type = Read

GPIDADR = 1

D15-D0
REV

REV: 4 digit BCD value of the FPGA code revision

READ_UCNTR
Type = Read

GPIDADR = 2

D15-D0
PCNT

PCNT: This is a 16-bit pointer defining where the READOUT Processor is currently saving data. This value is a 16-word page count. The total memory address range is 0 to 0xFFFFF. This equates to 0 to 0xFFFF pages.

Note: To stream data from memory in real time, the user must poll this register to keep track of where the READOUT Processor is saving data and therefore how much data can be downloaded.

READ_SID0, 1, 2, 3
Type = Read

GPIDADR = 3, 4, 5, 6

D15-D0
ID DATA

These four ports form the 64-bit ID of the Matrix-EVB board as read from the SERIAL ID chip. The first word is the most significant word. The fourth word is the least significant word.

FIFO Memory

The 2Mbytes of memory are configured as 1M x 16-bit words. When the readout process is running, the FPGA streams the event data from the 16 channels to this memory in consecutive address locations. When the maximum address is reached the address will wrap-round and storing will begin at the base address. Hence the memory can be treated as a 1M x 16-bit FIFO. The FIFO provides a 16-bit register that continually reports where the READOUT Processor is currently saving data. This value is a 16-bit word page count. The total memory address range is 0 to 0xFFFFF. This equates to 0 to 0xFFFF pages. To stream data from memory in real time, the user must poll this register to keep track of where the READOUT Processor is saving data to determine how much data can be downloaded.

Readout FIFO Data Format

When the Matrix-EVB Board is commanded to begin collecting readout data, the FPGA polls the 16 channels for messages conforming to the Readout Data Stream format. On receipt of a message the data is pushed to a FIFO residing in the 2MByte memory block. Using the USB interface, the host computer can download this FIFO data in real time to memory or file. The format of the FIFO data will vary depending on the mode of operation.

MODE 0

Each event will be represented by 4 x 16-bit data words. These words will be saved in the FIFO.

WNo.	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
W0	1	1	1	1	0	0	0	0	0	0	0	0	C	C	C	C
W1	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
W2	X	1	0	0	M	M	M	M	A	A	A	A	P	P	P	P
W3	0	0	0	0	E	E	E	E	E	E	E	E	E	E	E	E

The first word will always have bits 15 to 12 set to '1'.

C: Readout channel number (0 – 15)

T: 16-bit TDC value with 500ps LSB

X: This bit is set if the data is invalid. If this bit is '1' the data must be ignored.

M: Operating Mode

A: Array number (0-8)

P: Pixel number (0-15)

E: 12-bit energy value

MODE 1

Each event will be represented by 12 x 16-bit data words. These words will be saved in the FIFO.

WNo.	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
W0	1	1	1	1	0	0	0	0	0	0	0	0	C	C	C	C
W1	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
W2	X	1	0	0	M	M	M	M	A	A	A	A	P	P	P	P
W3	H	H	H	H	E0	E0	E0	E0	E0	E0	E0	E0	E0	E0	E0	E0
W4	H	H	H	H	E1	E1	E1	E1	E1	E1	E1	E1	E1	E1	E1	E1
|	|	|	|	|	|	|	|	|	|	|	|	|	|	|	|	|
W11	H	H	H	H	E8	E8	E8	E8	E8	E8	E8	E8	E8	E8	E8	E8

The first word will always have bits 15 to 12 set to '1'.

C: Readout channel number (0 – 15)

T: 16-bit TDC value with 500ps LSB

X: This bit is set if the data is invalid. If this bit is '1' the data must be ignored.

M: Operating Mode

A: Array number (0-8)

P: Pixel number (0-15)

E0: 12-bit energy value of EVENT Pixel

E1-E8: 12-bit energies from surrounding pixels

H: If the energy is not on array due to boundary pixel, these bits will be set HI and the energy will always be 0. If the energy is real then these bits will always be 0.

The relationship between energy number and surrounding pixels is as follows:

E1	E2	E3
E8	E0	E4
E7	E6	E5

MODE 2

Each event will be represented by 19 x 16-bit data words. These words will be saved in the FIFO.

WNo.	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
W0	1	1	1	1	0	0	0	0	0	0	0	0	C	C	C	C
W1	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
W2	X	1	0	0	M	M	M	M	A	A	A	A	P	P	P	P
W3	0	0	0	0	E0	E0	E0	E0	E0	E0	E0	E0	E0	E0	E0	E0
W4	0	0	0	0	E1	E1	E1	E1	E1	E1	E1	E1	E1	E1	E1	E1
|	|	|	|	|	|	|	|	|	|	|	|	|	|	|	|	|
W18	0	0	0	0	E15	E15	E15	E15	E15	E15	E15	E15	E15	E15	E15	E15

The first word will always have bits 15 to 12 set to '1'.

C: Readout channel number (0 – 15)

T: 16-bit TDC value with 500ps LSB

X: This bit is set if the data is invalid. If this bit is '1' the data must be ignored.

M: Operating Mode

A: Array number (0-8)

P: Pixel number (0-15)

E0-E15: 12-bit energies of all ADCs. The energy values E0 – E15 correspond to the pixel position numbering as given in Figures 3 and 4 on pages 9 and 10.

MODE 8

This MODE is reserved for reading back the READOUT board FPGA REV, ID and TEMPERATURE. When in this mode the READOUT board will send 1 message only. The format of this message is as follows:

Word 0: **0xF00X** where '**X**' is the channel number 0-15

Word 1: 4 digit BCD value of the FPGA code revision

Word 2-5: 64-bit board ID

Word 6: Temperature of sensor head

Word 7-15: Null (For future development)

Note: Word 6 is a signed number (16 bits), where 1 = 1/16°C. The table below shows some examples of the value to temperature conversion.

TEMPERATURE	DIGITAL OUTPUT (Binary)	DIGITAL OUTPUT (Hex)
+125°C	0000 0111 1101 0000	07D0h
+85°C*	0000 0101 0101 0000	0550h
+25.0625°C	0000 0001 1001 0001	0191h
+10.125°C	0000 0000 1010 0010	00A2h
+0.5°C	0000 0000 0000 1000	0008h
0°C	0000 0000 0000 0000	0000h
-0.5°C	1111 1111 1111 1000	FFF8h
-10.125°C	1111 1111 0101 1110	FF5Eh
-25.0625°C	1111 1110 0110 1111	FE6Fh
-55°C	1111 1100 1001 0000	FC90h

*The power on reset value of the temperature register is +85°C

For full details please refer to the DS1822 data sheet.

Configuration Commands

The FPGA, under command from the USB, configures the modules using a serial data line. The format of this serial data is RS232 with the following settings:

Signal Level	LVTTL
Baud Rate	115200
Start Bits	1
Stop Bits	1
Data Bits	8
Parity Bits	0

Configuration commands are passed to the **SET_SERPT** register in the FPGA as a series of ASCII characters. The characters are encoded into RS232 format serial data which is then passed on to the relevant module.

Set ARRAY

Format: **AX**

The message identifier is the character **A**. This shall be followed by a single character defining the ARRAY type (MatrixSM-9 or MatrixSM-9X1). This single character (X) shall be of value 0-1 defining the ARRAY type. 0 = MatrixSM-9, 1 = MatrixSM-9X1, 1=MatrixB-9X1.

Example: A1 Define READOUT module as MatrixSM-9X1.

Write to DAC

Format: **DXVVV**

The message identifier is the character **D**. This shall be followed by a single character (X) of value 0-5 defining the DAC number to write to. Three more characters (VVV) shall define a 3 digit HEX number to be written to the DAC.

Example: D3CEF Write 0xCEF to DAC 3.

DAC0 - ARRAY Threshold

DAC1 - ARRAY Offset

DAC2 - PIXEL Threshold

DAC3 - PIXEL Offset

DAC4 - BIAS Offset

DAC5 - ADC Offset

Set MODE

Format: **MX**

The message identifier is the character **M**. This shall be followed by a single character (X) of value 0-9 defining the mode to put the READOUT module in.

Example: M2 Put READOUT module in MODE 2.

Set READOUT

Format: **RX**

The message identifier is the character **R**. This shall be followed by a single character (X) of value 0-1. The character '1' will place the READOUT module into RESET. The character '0' will enable the module allowing it to search for events.

Example: R1 Put READOUT module into RESET state.

Set TEST LED

Format: **TX**

The message identifier is the character **T**. This shall be followed by a single character (X) of value 0-1 putting the READOUT module TEST LED ON or OFF.

Example: T1 Put READOUT TEST LED on.

Set SAMPLE TIME

Format: **SXX**

The message identifier is the character **S**. This shall be followed by 2 HEX characters (XX). This value shall be the delay time to sample the ADC energy after the event occurs. This value has an LSB of 10ns.

Example: S21 Delay is 21 Hex = 33 Decimal. Delay time = 33 x10ns = 330ns.

Set ENERGY LIMIT

Format: **LXXX**

The message identifier is the character **L**. This shall be followed by 3 HEX characters (XXX). On detecting a PIXEL EVENT the FPGA will compare the energy value with this LIMIT value. IF the value is less then the EVENT will be ignored. This value is a HEX number defining an ADC BIN value. This feature is used to filter EVENTS due to noise and hence improve the bandwidth of the system.

Example: L02F Ignore all events with an ADC value of less than 0x02F

Reset PORT

Format: **ZZZZZ**

Sending 5 consecutive 'Z' characters will reset the READOUT module serial port ready for a new message. This message is only used for special conditions such as when a previous command was aborted.

Example: ZZZZZ Reset the READOUT RS232 port.

Matrix Software Drivers

DLL DRIVER FUNCTIONS

Conventions

The Matrix GUI calls a set of functions in a DLL (**LeeMatrixAPI.DLL**) allowing easy configuration and transfer of data via the USB interface. The DLL Library uses data types throughout as follows:

Sbyte	Signed 8 bits (char)
Ubyte	Unsigned 8 bits
Sword	Signed 16 bits
Uword	Unsigned 16 bits
Slong	Signed 32 bits
Ulong	Unsigned 32 bits
Error	Unsigned 16 bits
LEE_HANDLE	Signed 32 bits (HANDLE)

These types and all relevant parameter constants for the drivers are defined in **LeeMatrixAPI.H**.

Except for **MX_Find( )**, all drivers return an Error value reporting the result of the function call.

The error value will be one of the following:

MNEMONIC	VALUE	DESCRIPTION
E_NO_ERROR	0	No error
E_FIND_MODULE_ERROR	1	Could not find MATRIX module
E_OPEN_MODULE_ERROR	2	Could not open session with MATRIX module
E_SETTINGS_ERROR	3	Error writing settings to USB port
E_CLOSE_MODULE_ERROR	4	Could not close session with MATRIX module
E_VERSION_ERROR	5	Invalid USB driver version
E_COMMAND_ERROR	6	Error while writing command to USB port
E_DATA_ERROR	7	Error while reading/writing data from/to the USB
E_SPI_RESPONSE_ERROR	8	SPI interface did not return correct value
E_FIFO_ERROR	9	FIFO over-run error

These error values are defined in **LeeMatrixErrs.H.**

Driver Descriptions

Driver - LEE_Find

LEE_API Ulong WINAPI LEE_Find (void)

This function returns the number of Matrix modules present on the USB bus.

Example:

```
#include “LeeMatrixAPI.h”
Ulong ModuleCount;
    ModuleCount = LEE_Find();
    If(!ModuleCount)
        printf(“No Module Found”);
```

Driver - LEE_Open

LEE_API Error WINAPI LEE_Open (LEE_HANDLE *MHandle, Uword ModNo)

This function opens communication session with the Matrix system.

ModNo: Module number.

MHandle: Pointer to an uninitialized handle.

The module number is 0 to (n-1) where n is the number of modules detected by LEE_Find().

On success E_NO_ERROR is returned and the handle is initialized.

Example:

```
#include “LeeMatrixAPI.h”
Ulong          ModuleCount;
Error          error;
LEE_HANDLE   mHandle;
    ModuleCount = LEE_Find();
    If(ModuleCount)
        Error = LEE_Open( &mHandle, 0);
    else
        printf(“No Module Found”);
```

Driver – LEE_Close

LEE_API Error WINAPI LEE_Close (LEE_HANDLE *MHandle)

This function closes the current communication session with the Matrix system.

MHandle: Session handle.

Driver – LEE_Wdata

LEE_API Error WINAPI LEE_Wdata (LEE_HANDLE *MHandle, Uword *data, Ulong wCnt)

This function streams a data block from the host PC to the system.

MHandle: Session handle.

data: Pointer to array containing the data

wCnt: Number of words to transfer

Driver – LEE_Rdata

LEE_API Error WINAPI LEE_Rdata (LEE_HANDLE *MHandle, Uword *data, Ulong wCnt)

This function streams a data block from the system to the host PC.

MHandle: Session handle.

data: Pointer to array where data is to be stored.

wCnt: Number of words to transfer.

Driver – LEE_DLL_Version

LEE_API Uword WINAPI LEE_DLL_Version (void)

The returned value will be a 4 digit BCD value defining the version of the DLL.

Example 0x0112 = Version 1.12

Driver - LEE_TDCreset

LEE_API Error WINAPI LEE_TDCreset (LEE_HANDLE MHandle, Uword State)

This function sets the TDC reset signal on the Matrix-EVB Board to the value defined by 'State'.

MHandle: Session handle.

State: Set TRUE to reset the TDCs. Set FALSE to run the TDCs.

> **Note:** The TDC reset line is a global signal to all Readout Module TDCs in the system.

Driver - LEE_SetMode

LEE_API Error WINAPI LEE_SetMode (LEE_HANDLE MHandle, Uword Mode)

This function sets the mode for the readout.

MHandle: Session handle.

Mode: Mode to set.

ID Name (In header file)	Code Value	Operating mode
SINGLE_MODE	0	Single
REGION_MODE	1	Region of Interest
ALL_ENERGY_MODE	2	All Energies
READOUT_INFO_MODE	8	Get Readout Info

> **Note:** This is a global definition. The individual Readout modules must not be programmed to operate in different modes.

Driver - LEE_SetReadoutLIN9

Uword WINAPI LEE_SetReadoutLIN9 (LEE_HANDLE MHandle, Uword Channel, Uword State)

This function sets the selected module in MatrixSM-9 or MatrixSM-9X1 mode.

MHandle: Session handle.

Channel: Module cannel (0-15)

State: 0 = MatrixSM-9, 1 = MatrixSM-9X1, 1=MatrixB-9X1

Driver - LEE_SetSampleCount

LEE_API Error WINAPI LEE_SetSampleCount (LEE_HANDLE MHandle,

Uword Channel, Uword Count)

This function sets the sampling time between when an event occurs and when the ADC takes the energy sample value.

MHandle: Session handle.

Channel: Module cannel (0-15)

Count: Count time.

Note: The Count is a value in the range 0-255 defining the time from 0μS to 2.55μS (LSB = 10ns).

Driver - LEE_SetEnergyLimit

LEE_API Error WINAPI LEE_SetEnergyLimit (LEE_HANDLE MHandle,

Uword Channel, Uword Energy)

This function sets the minimum allowable energy value for an event.

MHandle: Session handle.

Channel: Module cannel (0-15)

Count: Count time.

This value defines the minimum energy allowable for an event to be regarded as true. This value is input as an ADC bin number. When an event is detected, the ADC value for the pixel is compared with this value. If it is less than the **Energy Limit** value the event will be discarded. This is used to filter out undesirable noise related events and hence improve the overall event bandwidth.

Driver - LEE_SetUsbConReg

LEE_API Error WINAPI LEE_SetUsbConReg(LEE_HANDLE MHandle, Uword Value)

This function sets the USB control register to a specific value.

MHandle: Session handle.

Value: Value to set register to.

Driver - LEE_GetUsbConReg

LEE_API Error WINAPI LEE_GetUsbConReg(LEE_HANDLE MHandle, Uword *Value)

This function reads back the current value of the USB control register.

MHandle: Session handle.

Value: Pointer to location for storing the current value.

Driver - LEE_SetConfigReg

LEE_API Error WINAPI LEE_SetConfigReg(LEE_HANDLE MHandle, Uword Value)

This function sets the USB Configuration register to a specific value.

MHandle: Session handle.

Value: Value to set register to.

Driver - LEE_SetCountHiReg

LEE_API Error WINAPI LEE_SetCountHiReg(LEE_HANDLE MHandle, Uword Value)

This function sets the USB Counter HI register to a specific value.

MHandle: Session handle.

Value: Value to set register to.

Driver - LEE_SetCountLoReg

LEE_API Error WINAPI LEE_SetCountLoReg(LEE_HANDLE MHandle, Uword Value)

This function sets the USB Counter LO register to a specific value.

MHandle: Session handle.

Value: Value to set register to.

Driver - LEE_ReadUsbStatus

LEE_API Error WINAPI LEE_ReadUsbStatus(LEE_HANDLE MHandle, Uword *Value)

This function reads the value of the USB status register.

MHandle: Session handle.

Value: Pointer to location for storing value.

Driver - LEE_ReadFpgaRev

LEE_API Error WINAPI LEE_ReadFpgaRev(LEE_HANDLE MHandle, Uword *Value)

This function reads the value of the Matrix-EVB Board FPGA revision.

MHandle: Session handle.

Value: Pointer to location for storing value.

This will be a 4-digit BCD value defining the revision of the FPGA code.

Example: 0x0112 = Rev 1.12

Driver - LEE_ReadCounterReg

LEE_API Error WINAPI LEE_ReadCounterReg(LEE_HANDLE MHandle, Uword *Value)

This function reads the value of the FIFO counter register.

MHandle: Session handle.

Value: Pointer to location for storing value.

Note: The Counter Register is a page count defining the current position of the data pointer in the FIFO. The page size is 16 words. Therefore a value of 0x8100 corresponds to an address of 0x81000 in the FIFO memory.

Driver - LEE_ReadSerialID

LEE_API Error WINAPI LEE_ReadSerialID(LEE_HANDLE MHandle, Uword *Value)

This function reads the 64-bit serial ID number of the Matrix-EVB Board.

MHandle: Session handle.

Value: Pointer to a 4 x 16-bit array for storing the value.

Note: The value of the 64-bit ID will be stored in 'Value'. The MSB of Value[0] will be the MSB of the 64-bit ID and the LSB of Value[3] will be the LSB of the 64-bit ID.

Driver - LEE_SetLED

LEE_API Error WINAPI LEE_SetLED (LEE_HANDLE MHandle, Uword State)

This function sets the state of the Matrix-EVB Board test LED.

MHandle: Session handle.

State: Set TRUE for ON, Set FALSE for OFF.

Driver - LEE_Set ReadoutLED

LEE_API Error WINAPI LEE_SetReadoutLED (LEE_HANDLE MHandle,
Uword Channel, Uword State)

This function sets the state of the selected Readout Module test LED.

MHandle: Session handle.

Channel: Channel number.

State: Set TRUE for ON, Set FALSE for OFF.

Driver - LEE_ClearUsbAddress

LEE_API Error WINAPI LEE_ClearUsbAddress (LEE_HANDLE MHandle)

This function sets the USB FIFO address pointer to 0.

MHandle: Session handle.

Driver - LEE_ResetSerialPort

LEE_API Error WINAPI LEE_ResetSerialPort (LEE_HANDLE MHandle, Uword Channel)

This function resets the serial port of the specified channel.

MHandle: Session handle.

Channel: Channel number.

> **Note:** The configuration of the modules is carried out via a single serial port line. Messaging to this port can be aborted at any time by issuing a reset command. Channel must be set to 0-15 for a specific channel number or set to BROADCAST_CHANNEL to simultaneously reset all ports.

Driver - LEE_StartReadout

LEE_API Error WINAPI LEE_StartMatrix (LEE_HANDLE MHandle)

This function starts the Readout process running.

MHandle: Session handle.

Driver - LEE_SetDAC

LEE_API Error WINAPI LEE_SetDACS (LEE_HANDLE MHandle, Uword Channel, Uword *Values)

This function sets all the DACS of a specific channel to the values defined in the array Values.

MHandle: Session handle.

Channel: Channel number.

Values: Pointer to an array of 6 DAC values. See Note.

> **Note:** The position of the DAC values in the array is defined by the DAC ID constants. For example, the **Array Threshold DAC** will bet programmed with the value in Values [ARRAY_THRESHOLD_DAC].

ID Name (In header file)	Code Value	DAC
ARRAY_THRESHOLD_DAC	0	Array Threshold DAC
ARRAY_REFERENCE_DAC	1	Array DC Offset DAC
PIXEL_THRESHOLD_DAC	2	Pixel Threshold DAC
PIXEL_REFERENCE_DAC	3	Pixel DC Offset DAC
BIAS_OFFSET_DAC	4	Bias voltage setting DAC
ADC_REFERENCE_DAC	5	ADC DC Baseline Offset DAC

Driver - LEE_ConfigureDACS

LEE_API Error WINAPI LEE_ConfigureDACS (LEE_HANDLE MHandle, Uword Mode,
Uword *Bias, Uword *AdcR,
Uword *PixelT, Uword *PixelR,
Uword *ArrayT, Uword *ArrayR)

This function loads the DACS with the values defined by the input parameters.

MHandle: Session handle.

Mode: Mode for setting up DACS (See note below)

Bias: Pointer to array of 16 SiPM bias values (0-4095).

AdcR: Pointer to array of 16 ADC DC baseline offset values (0-4096).

PixelT: Pointer to array of 16 Threshold values for the pixels (0-4095).

PixelR: Pointer to array of 16 DC offset values for the pixels (0-4095).

ArrayT: Pointer to array of 16 Threshold values for the arrays (0-4095).

ArrayR: Pointer to array of 16 DC offset values for the arrays (0-4095).

The mode must be set to **NOT_BROADCAST** or **BROADCAST_CHANNEL**. If the mode is set to NOT_BROADCAST then all the modules will be configured with the 16 different values passed in the arrays. For example, Channel 0 bias will be set to Bias[0], Channel 1 bias will be set to Bias[1] etc.

If the mode is set to BROADCAST_CHANNEL then all modules will be configured with the same values as defined in the first element of the array. For example, the bias for all channels will be set to the value defined in Bias [0].

Driver - LEE_ResetUSB

LEE_API Error WINAPI LEE_ResetUSB (LEE_HANDLE MHandle)

This function resets the USB process. After calling this function the system will stop streaming data from the FIFO to the USB bus.

MHandle: Session handle.

Driver - LEE_InitUSB

LEE_API Error WINAPI LEE_InitUSB (LEE_HANDLE MHandle, Uword Clear, Ulong Count)

This function initializes the USB interface for a block transfer.

MHandle: Session handle.

Clear: Set TRUE to clear the USB FIFO address pointer. Set FALSE for no change.

Count: Number of 16-bit words to transfer.

The transfer will start at the current address defined by the FIFO address pointer. If 'Clear' is set TRUE the address pointer will be cleared and the transfer will start at the first location in the FIFO. Normally this function should be called once with 'Clear' set TRUE when the system readout is started. Further calls to this function should have 'Clear' set to 0 to allow further data to be read from the FIFO as it arrives.

Driver - LEE_ReadoutReset

LEE_API Error WINAPI LEE_ReadoutReset (LEE_HANDLE MHandle, Uword State)

This function sets the Matrix-EVB Board Readout Processor to the value defined by 'State'.

MHandle: Session handle.

State: Set TRUE to reset the Readout Processor. Set FALSE to run the Processor.

Driver - LEE_ChannelReadoutReset

LEE_API Error WINAPI LEE_ChannelReadoutReset (LEE_HANDLE MHandle,

Uword Channel, Uword State)

This function sets the specified Readout Module Readout Processor to the value defined by 'State'.

MHandle: Session handle.

Channel: Readout Module channel number.

State: Set TRUE to reset the Readout Processor. Set FALSE to run the Processor.

> **Note:** Channel must be set to 0-15 for a specific channel number or set to **BROADCAST_CHANNEL** to simultaneously set all channels to the same value.

Driver - LEE_GetREADOUTinfo

LEE_API Error WINAPI LEE_GetREADOUTinfo(LEE_HANDLE MHandle,

Uword Mode, Uword *buff)

This function reads the 64-bit serial ID number of the Matrix-EVB Board.

MHandle: Session handle.

Mode: Mode to put readout module back to once function is completed.

buff: Pointer to 256 x 16-bit array for storing the module(s) info

This function gets the info for all modules (up to 16) simultaneously. The data will be stored in the 256 word array in 16 word blocks. The info for module 0 will be stored in buff[0-15]. The info for module 15 will be stored in buff[240-255]. The format of this data will be as follows:

Word 0: 4 digit BCD value of the FPGA code revision

Word 1-4: 64-bit board ID

Word 5*: Temperature of sensor head

Upper byte	Temp HIGH byte
Lower byte	Temp LOW byte

Word 6-15: Null (For future development)

For details of byte values see DS1822 specification document.

If the first element in the module's info (FPGA code revision) is 0 then no module was detected.

Driver - LEE_GetFIFOdata

LEE_API Error WINAPI LEE_GetFIFOdata(LEE_HANDLE MHandle,

Uword *buff, Ulong *Count)

This function reads into a buffer all data that has been placed in the FIFO since the last call to this function.

MHandle: Session handle.

buff: Pointer to buffer for storing FIFO data

Count: Pointer to location for storing the number of words received in buff

On completion, the value of **Count** will be the total number of 16-bit words copied from the FIFO to the **buff**. The format of this data will be as described in **FIFO Data Format**. However, due to a feature of the FIFO streaming, each word is repeated in the FIFO. For example, a single event will be stored in the FIFO as 8 x 16-bit words W0, W0, W1, W1, W2, W2, W3 and W3.

- This function implements an address pointer for keeping track with the FPGA page register **(READ_UCNTR)**
- When the readout sequence is started this pointer is automatically set to 0.
- When this function is called, the **READ_UCNTR** is compared with the address pointer to determine how many words are to be transferred.
- Once the data is transferred, the pointer is updated with the value of the **READ_UCNTR** register.
- Hence, consecutive calls to this function will automatically read the FIFO data.

Note: This function interrogates the MERR bit in the READ_USTAT register to ensure that a wrap-round error has not occurred. It is the responsibility of the user to ensure that the repetition rate of this function is sufficient to ensure the page pointer is kept up with.

Example

This example puts the system in single energy mode, starts the process and streams the first 64K words of data to a memory buffer.

```
#include “LeeMatrixAPI.h”

Ulong         ModuleCount;
Ulong       buffSZ, count, total;
```

```
Error         error;
LEE_HANDLE    mHandle;
Uword        *buffer;


    /*
     Set buffer size to 1Meg.
    */
    buffer = NULL;
    buffSZ = 0x100000;
    /*
     Get the number of modules on USB bus.
    */
    ModuleCount = LEE_Find();
    /*
     If a module exists then open comms with first module.
    */
    if(ModuleCount)
      error = LEE_Open(&mHandle, 0);
  else
      error = E_FIND_MODULE_ERROR;
    /*
     Allocate the memory buffer.
    */
    if(error == E_NO_ERROR)
    {
        buffer = (Uword *)malloc(buffSZ);
        if(mBuffer == NULL)
            error = E_MALLOC_ERROR;
    }
    /*
```

```
 Set the mode for single energy events.
*/
if(error == E_NO_ERROR)
    error = LEE_SetMode(mHandle, SINGLE_MODE);
/*
 Start READOUT process running.
*/
if(error == E_NO_ERROR)
    error = LEE_StartReadout(mHandle);
/*
 Now stream the FIFO to buffer until 64K words have been received.
*/
for(total=0; total < 0x10000 && error == E_NO_ERROR; total += count)
    error = LEE_GetFIFOdata(MHandle, &buffer[total], &count);
/*
 Free the buffer.
*/
if(buffer)
    free(buffer);
/*
 Print possible error.
*/
if(error == E_NO_ERROR)
    printf("Success!");
else
    printf("Error encountered: %d", error);
```

Appendix

TIMING DATA PROCESSING

When running a timing plot, a binary format **TIM** file will be created that will record all events. The format of this file is described on pages 34 - 35. This file can be used to determine the Timing Resolution by the following steps:

Sort Coincident Pairs

During a Timing Plot, the Filter on the Main GUI page is checked. Therefore it can be assumed that the delta time between two consecutive events is the difference between the timestamps. By reading through the **TIM** file, coincident pairs and singles can be determined based on the time difference and a user defined limit for a coincidence.

In its simplest form the algorithm could be as the follows pseudo code:

```
IF (Channel of Event A ≠ Channel of Event B and Delta Time < Limit)
{
    Events are a pair
}
```

However, this algorithm can be extended to consider groups of events (e.g. four consecutive events at a time) to avoid the possibility of creating false pairs.

This will provide a new '**modified TIM**' file with Event Pairs only.

Time-walk Correction

The timing of the events is determined by the output of voltage comparators. The output of these comparators, also known as 'leading edge discriminators', change logic level when the input voltage exceeds a specified threshold. Figure 17 shows as simplified drawing of two coincident pulses arriving at two different channels. In this example the event on channel A has a greater peak voltage than the event on channel B. It can be seen that this results in the comparator threshold being reached at different times creating a timing error of dT. This error is known as the *time-walk* and is a well-documented problem with leading edge discriminators.

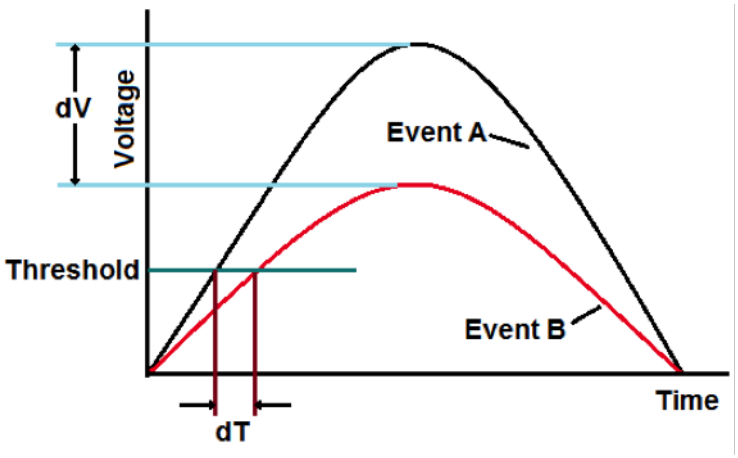


Figure 17

To avoid this problem the hardware solution is to use a Constant Fraction Discriminator (CFD) that resolves the start time of the signal. However, CFDs are complicated, incorporating several components. Also, the CFD will not account for noise errors where two identical signals, to the same channel, trigger at different times due to the noise level at that particular time.

To account for these errors it is necessary to post process the data in software. It can be seen in Figure 17 that the value of dT will vary with the difference in amplitude dV. This relationship between dT and dV is linear. The Matrix9 energy value, reported by an event, is the ADC value of the output of an integrator sampled at a specific time after the input pulse is detected. Therefore this value, dE, will be proportional to the peak value of the input pulse.

Time-walk compensation by amplitude is a standard, commonly accepted method. The time difference between two coincident events is adjusted based on the energy difference of those events. Therefore the process to compensate for time-walk is as follows:

1. Calculate Slope

Using the '**modified TIM**' file to produce a set of points dT v dE where dT and dE are the difference in Timestamp and Energy value respectively for each pair of events.

Example:

Event A: T = 0x8010 E = 0x0080

Event B: T = 0x8090 E = 0x0040

dT = -0x80 dE = +0x40

Note:

- Subtract the same channel from the other regardless of the order in the TIM file.
- Always preserve the sign

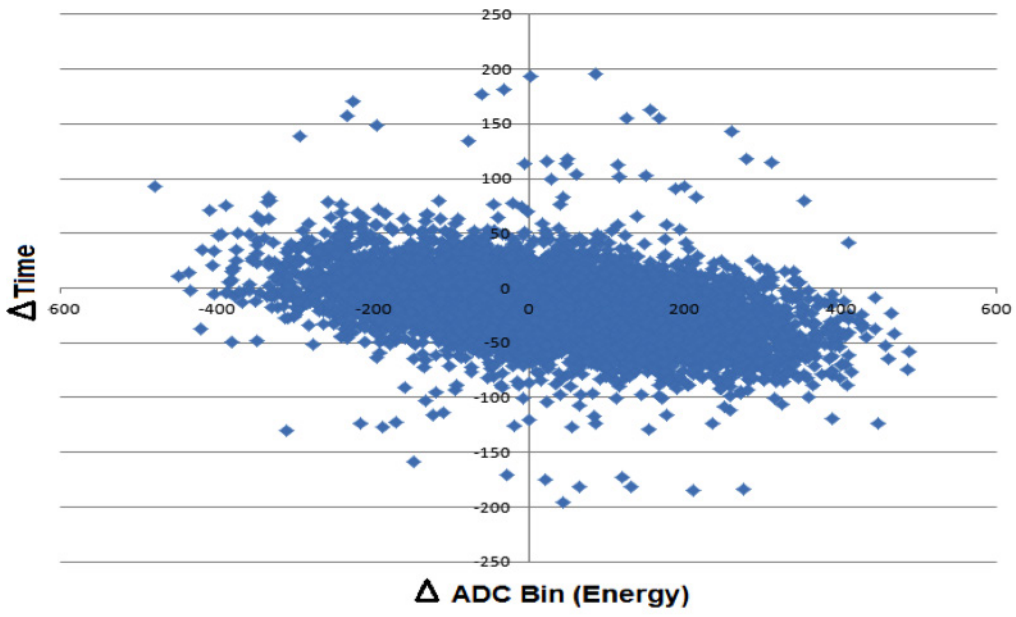


Figure 18

From this set of points calculate the best fit. Figure 18 shows a scatter graph of a typical set of data. It can be seen from the scatter graph that there is a relationship between the Time and Energy. The processing now determines the best fit slope of the data. To avoid extreme dT points influencing the result, these points should be removed before the slope is calculated.

2. Adjust the Timing

Once the slope is determined the delta time between pairs is adjusted as follows:

$$dT(adjusted) = dT - (Slope \times dE)$$

3. Calculate the Timing Resolution

Once the new dT values are determined the values are binned (each bin = 500ps) to provide histogram data. From this histogram a region of interest is determined for calculating the best Gaussian fit. The Timing Resolution is the FWHM value of this Gaussian fit.

SensL © 2012